IBM MQ

Installing IBM MQ

Version 9 Release 0

Note

Before using this information and the product it supports, read the information in “Notices” on page 1629.

This edition applies to version 9 release 0 of IBM MQ and to all subsequent releases and modifications until otherwise indicated in new editions.

Contents

Figures

Tables

Planning an IBM MQ architecture

When planning your IBM® MQ environment, consider the support that IBM MQ provides for single and multiple queue manager architectures, and for point-to-point and publish/subscribe messaging styles. Also plan your resource requirements, and your use of logging and backup facilities.

Before you plan your IBM MQ architecture, familiarize yourself with the basic IBM MQ concepts. See IBM MQ Technical overview.

IBM MQ architectures range from simple architectures using a single queue manager, to more complex networks of interconnected queue managers. Multiple queue managers are connected together using distributed queuing techniques. For more information about planning single queue manager and multiple queue manager architectures, see the following topics:

- "Architectures based on a single queue manager" on page 5
- "Architectures based on multiple queue managers" on page 6
  - "Planning your distributed queues and clusters" on page 7
  - "Planning your distributed publish/subscribe network" on page 61

z/OS On IBM MQ for z/OS® you can use shared queues and queue-sharing groups to enable you to implement workload balancing, and your IBM MQ applications to be scalable and highly available. For information about shared queues and queue-sharing groups, see Shared queues and queue-sharing groups.

IBM MQ provides two different release models:

- The Long Term Support (LTS) release is most suitable for systems requiring a long term deployment and maximum stability
- The Continuous Delivery (CD) release is intended for systems which need to rapidly exploit the latest functional enhancements for IBM MQ.

Both release types are installed in the same way, but there are considerations relating to support and migration which you need to understand. See "IBM MQ release types" for further information.

For information about planning for multiple installations, storage and performance requirements, and use of clients, see the other subtopics.

Related concepts:

z/OS "Planning your IBM MQ environment on z/OS" on page 138
When planning your IBM MQ environment, you must consider the resource requirements for data sets, page sets, DB2®, Coupling Facilities, and the need for logging, and backup facilities. Use this topic to plan the environment where IBM MQ runs.

Related information:

Checking requirements

Making sure that messages are not lost (logging)

Availability, recovery and restart

IBM MQ release types

V 9.0.0

From IBM MQ Version 9.0 there are two types of release; a Long Term Support (LTS) release and Continuous Delivery (CD) release. The aim of the two release types is to meet the requirement for delivery of new and enhanced IBM MQ functions as rapidly as possible in the next CD release, while at the same time maintaining a stable long term support release for systems that need a long term deployment of IBM MQ, and for customers who prefer this traditional option.

The release model

The two release types are distinguishable by the modification number in the version.release.modification (`v.r.m`) release identifier.

IBM MQ Version 9.0.0 has a zero modification number and is designated as LTSR.

CD Continuous Delivery releases have a non-zero modification number, for example, 9.0.1, 9.0.2, and so on.

CD releases are made available on a regular basis, and contain *functional enhancements* as well as the latest set of defect fixes and security updates.

Each CD release replaces the prior ones for that version of IBM MQ, so it is possible for you to skip CD releases, if a specific CD release does not contain any function that is relevant for your enterprise. However, fixes will only be provided on the latest two CD releases.

LTS LTS releases are updated by fix packs or PTFs, which provide defect fixes and security updates in a predictable fashion. This process matches the way service has been provided on prior versions of IBM MQ. Fix packs are numbered according to a `v.r.m.f` scheme so, for example, the first fix pack for the IBM MQ Version 9.0.0 LTS release will be numbered 9.0.0.1.

For more information, see IBM MQ FAQ for Long Term Support and Continuous Delivery releases.

The next sections describe the specific detail of the process for obtaining, installing, and using IBM MQ code in the new release model for IBM MQ for z/OS and IBM MQ for Multiplatforms.

z/OS
Considerations for IBM MQ for z/OS

Ordering

When ordering IBM MQ for z/OS Version 9.0, two separate features are offered on ShopZ. The features correspond to the LTS release and the CD release.

Both features are applicable to the same product ID (PID). It is the product ID that is licensed, so where one feature is licensed, there is entitlement to use the alternative feature if required.

When ordering the following IBM MQ for z/OS products, select the feature corresponding with either the LTS release or CD release:
- IBM MQ for z/OS Version 9.0
- Managed File Transfer for z/OS Version 9.0

If you are selecting products for inclusion in a ServerPac, it is not possible to choose both the LTS release and CD release in the same ServerPac order, because the products cannot be installed by SMP/E in the same target zone.

Advanced Message Security for z/OS (AMS) Version 9.0 provides both the LTS release and CD release options.

The release options are selected by choosing the equivalent release for the base product, IBM MQ for z/OS Version 9.0, and enabling with the product 5655-AM9.

Attention: You cannot choose CD release enhancements for AMS functions on an LTS release IBM MQ base.

If your enterprise uses IBM MQ for z/OS Value Unit Edition (VUE) Version 9.0, the same product 5655-VU9 enables VUE licensing in either the LTS release or CD release of IBM MQ for z/OS Version 9.0.

Installation

The LTS and CD releases are provided in separate sets of FMIDs. Note that these FMIDs cannot be installed in the same SMP/E target zone.

Where you require both the LTS and CD releases, you should:
- Install the LTS release and CD release in separate target zones
- Maintain separate target and distribution libraries for the two releases

As the releases use different FMIDs, it is not possible to update a CD release with maintenance for an LTS release or the other way round. Similarly, there is no way to switch a version of the product code from LTS release to CD release or the other way round.

However it is possible to switch a queue manager between the release models, for further information see Migration

Maintenance

The LTS release is serviced by the application of PTFs which provide defect fixes. Note that the PTFs resolve specific issues, and are not cumulative.

IBM provides groups of fixes in RSU levels for an LTS release.

The CD release is serviced by the application of PTFs which deliver both defect fixes and functional enhancements.

Each set of PTFs is cumulative and increases the modification level of the CD release. That is, a set of PTFs will update the product and change the reported `v.r.m` from V9.0.1 to V9.0.2. The subsequent set of PTFs supersedes the first set, and updates the installed product to V9.0.3.

Migration between LTS release and CD release

There are constraints and limitations, but generally a single queue manager can be migrated from using LTS release code to CD release code or from using CD release code to LTS release code provided that the target release is higher than that in use prior to the migration.

IBM MQ for z/OS has traditionally provided a fallback capability (backward migration) so that after a period of running following a migration, it is possible to fallback to the prior release without loss of data provided that new release operation is restricted to compatible operations through the OPMODE(COMPAT) control.

This capability is retained for LTS releases, but is not possible when the source or target of a migration is a CD release. The following are valid migration scenarios and illustrate this principle:

Source release	Destination release	Notes
V8.0.0	V9.0.0 LTSR	backward migration supported while V9.0.0 in OPMODE(COMPAT)
V8.0.0	V9.0.1 CDR	No backward migration, target release is CD
V9.0.0 LTSR	V9.0.2 CDR	No backward migration, target release is CD

Once a queue manager starts running a CD release of code, the queue manager sets the OPMODE to NEWFUNC at the new release level. The queue manager command level will also be updated to indicate the new release level.

Multi

Considerations for IBM MQ for Multiplatforms

Ordering

Within Passport Advantage there are two separate eAssemblies for IBM MQ Version 9.0. One contains installation images for IBM MQ Version 9.0.0 Long Term Support release, and the other contains installation images for IBM MQ Version 9.0.x Continuous Delivery release.

Download installation images from the eAssembly according to your choice of release.

Alternatively, for the LTS release only, a media pack is available, containing product installation DVDs.

All IBM MQ versions, and for IBM MQ Version 9.0 both the LTS releases and CD releases, belong to the same Product Id.

Entitlement to use IBM MQ extends across the entire product (PID) subject to the constraints of licensed components and pricing metrics. This means that you can freely choose between LTS release and CD release installation images for IBM MQ Version 9.0.

Installation

Once an installation image has been downloaded from Passport Advantage, you should only select for installation the components for which you have purchased entitlement. See IBM MQ license information for further information about which installable components are included for each chargeable component.

It is possible to install IBM MQ Version 9.0 LTS release and IBM MQ Version 9.0 CD release on the same operating system image.

If you do this, the components appear as separate installations, as supported by IBM MQ multi-version support. Each version has distinct sets of queue managers associated with that version.

Each new CD release is provided as an installation image. The new CD release can be installed along side an existing release, or, an earlier CD release can be updated in place by the installer to the new release.

Maintenance

The LTS release is serviced by the application of fix packs, which provide defect fixes. The fix packs are made available periodically and are cumulative.

The only maintenance provided for a CD release will be in the form of an iFix delivered to resolve a specific customer issue, if required, on the two latest CD releases, which could be on a subsequent version.

Migration between LTS release and CD release

There are constraints and limitations but, generally, a single queue manager can be migrated from using LTS release code to CD release code, or from using CD release code to LTS release code, provided that the target release is higher than that in use prior to the migration.

Two approaches are possible:
- Install the new release of code in place so that an existing installation of IBM MQ is updated. Any queue managers associated with the installation use the new release of code when started.
- Install the new release of code as a new installation, then move individual queue manager instances to the new installation using the setmqm command.

Once a queue manager starts running a CD release of code, the queue manager command level is updated to indicate the new release level. This means any new functions provided in the release are enabled, and it will no longer be possible to restart the queue manager using a code release with a lower `v.r.m`.

Architectures based on a single queue manager

The simplest IBM MQ architectures involve the configuration and use of a single queue manager.

Before you plan your IBM MQ architecture, familiarize yourself with the basic IBM MQ concepts. See IBM MQ Technical overview.

A number of possible architectures using a single queue manager are described in the following sections:
- “Single queue manager with local applications accessing a service”
- “Single queue manager with remote applications accessing a service as clients”
- “Single queue manager with a publish/subscribe configuration”

Single queue manager with local applications accessing a service

The first architecture based on a single queue manager is where the applications accessing a service are running on the same system as the applications providing the service. An IBM MQ queue manager provides asynchronous intercommunication between the applications requesting the service and the applications providing the service. This means that communication between the applications can continue even if one of the applications is offline for an extended period of time.

Single queue manager with remote applications accessing a service as clients

The second architecture based on a single queue manager has the applications running remotely from the applications providing the service. The remote applications are running on different systems to the services. The applications connect as clients to the single queue manager. This means that access to a service can be provided to multiple systems through a single queue manager.

A limitation of this architecture is that a network connection must be available for an application to operate. The interaction between the application and the queue manager over the network connection is synchronous.

Single queue manager with a publish/subscribe configuration

An alternative architecture using a single queue manager is to use a publish/subscribe configuration. In publish/subscribe messaging, you can decouple the provider of information from the consumers of that information. This differs from the point to point messaging styles in the previously described architectures, where the applications must know information about the target application, for example the queue name to put messages on. Using IBM MQ publish/subscribe the sending application publishes a message with a specified topic based on the subject of the information. IBM MQ handles the distribution of the message to applications that have registered an interest in that subject through a subscription. The receiving applications also do not need to know anything about the source of the messages to receive them. For more information, see Publish/subscribe messaging and Example of a single queue manager publish/subscribe configuration.

Related concepts:

"Planning an IBM MQ architecture" on page 1
When planning your IBM MQ environment, consider the support that IBM MQ provides for single and multiple queue manager architectures, and for point-to-point and publish/subscribe messaging styles. Also plan your resource requirements, and your use of logging and backup facilities.

Related information:

Introduction to IBM MQ

Creating and managing queue managers on Multiplatforms

Architectures based on multiple queue managers

You can use distributed message queuing techniques to create an IBM MQ architecture involving the configuration and use of multiple queue managers.

Before you plan your IBM MQ architecture, familiarize yourself with the basic IBM MQ concepts. See IBM MQ Technical overview.

An IBM MQ architecture can be changed, without alteration to applications that provide services, by adding additional queue managers.

Applications can be hosted on the same machine as a queue manager, and then gain asynchronous communication with a service hosted on another queue manager on another system. Alternatively, applications accessing a service can connect as clients to a queue manager that then provides asynchronous access to the service on another queue manager.

Routes that connect different queue managers and their queues are defined using distributed queuing techniques. The queue managers within the architecture are connected using channels. Channels are used to move messages automatically from one queue manager to another in one direction depending on the configuration of the queue managers.

For a high level overview of planning an IBM MQ network, see "Designing distributed queue manager networks" on page 8.

For information about how to plan channels for your IBM MQ architecture, see IBM MQ distributed queuing techniques.

Distributed queue management enables you to create and monitor the communication between queue managers. For more information about distributed queue management, see Introduction to distributed queue management.

Related concepts:

"Planning an IBM MQ architecture" on page 1
When planning your IBM MQ environment, consider the support that IBM MQ provides for single and multiple queue manager architectures, and for point-to-point and publish/subscribe messaging styles. Also plan your resource requirements, and your use of logging and backup facilities.

Related information:

Creating and managing queue managers on Multiplatforms

Planning your distributed queues and clusters

You can manually connect queues hosted on distributed queue managers, or you can create a queue manager cluster and let the product connect the queue managers for you. To choose a suitable topology for your distributed messaging network, you need to consider your requirements for manual control, network size, frequency of change, availability and scalability.

Before you begin

This task assumes that you understand what distributed messaging networks are, and how they work. For a technical overview, see Distributed queuing and clusters.

About this task

To create a distributed messaging network, you can manually configure channels to connect queues hosted on different queue managers, or you can create a queue manager cluster. Clustering enables queue managers to communicate with each other without the need to set up extra channel definitions or remote queue definitions, simplifying their configuration and management.

To choose a suitable topology for your distributed publish/subscribe network, you need to consider the following broad questions:

- How much manual control do you need over the connections in your network?
- How big will your network be?
- How dynamic will it be?
- What are your availability and scalability requirements?

Procedure

- Consider how much manual control you need over the connections in your network.

  If you only need a few connections, or if individual connections need to be very precisely defined, you should probably create the network manually.

  If you need multiple queue managers that are logically related, and that need to share data and applications, you should consider grouping them together in a queue manager cluster.
- Estimate how big your network needs to be.
  1. Estimate how many queue managers you need. Bear in mind that queues can be hosted on more than one queue manager.
  2. If you are considering using a cluster, add two extra queue managers to act as full repositories.

  For larger networks, manual configuration and maintenance of connections can be very time consuming, and you should consider using a cluster.
- Consider how dynamic the network activity will be.

  Plan for busy queues to be hosted on performant queue managers.

  If you expect queues to be frequently created and deleted, consider using a cluster.
- Consider your availability and scalability requirements.

1. Decide whether you need to guarantee high availability of queue managers. If so, estimate how many queue managers this requirement applies to.
2. Consider whether some of your queue managers are less capable than others.
3. Consider whether the communication links to some of your queue managers are more fragile than to others.
4. Consider hosting queues on multiple queue managers.

Manually configured networks and clusters can both be configured to be highly available and scalable. If you use a cluster, you need to define two extra queue managers as full repositories. Having two full repositories ensures that the cluster continues to operate if one of the full repositories becomes unavailable. Make sure that the full repository queue managers are robust, performant, and have good network connectivity. Do not plan to use the full repository queue managers for any other work.

- Based on these calculations, use the links provided to help you decide whether to manually configure connections between queue managers, or to use a cluster.

What to do next

You are now ready to configure your distributed messaging network.

Related information:

Configuring distributed queuing

Configuring a queue manager cluster

Designing distributed queue manager networks

IBM MQ sends and receives data between applications, and over networks using Queue Managers and Channels. Network planning involves defining requirements to create a framework for connecting these systems over a network.

Channels can be created between your system and any other system with which you need to have communications. Multi-hop channels can be created to connect to systems where you have no direct connections. The message channel connections described in the scenarios are shown as a network diagram in Figure 1 on page 9.

Channel and transmission queue names

Transmission queues can be given any name. But to avoid confusion, you can give them the same names as the destination queue manager names, or queue manager alias names, as appropriate. This associates the transmission queue with the route they use, giving a clear overview of parallel routes created through intermediate (multi-hopped) queue managers.

It is not so clear-cut for channel names. The channel names in Figure 1 on page 9 for QM2, for example, must be different for incoming and outgoing channels. All channel names can still contain their transmission queue names, but they must be qualified to make them unique.

For example, at QM2, there is a QM3 channel coming from QM1, and a QM3 channel going to QM3. To make the names unique, the first one might be named QM3_from_QM1, and the second might be named QM3_from_QM2. In this way, the channel names show the transmission queue name in the first part of the name. The direction and adjacent queue manager name are shown in the second part of the name.

A table of suggested channel names for Figure 1 on page 9 is given in Table 1 on page 9.

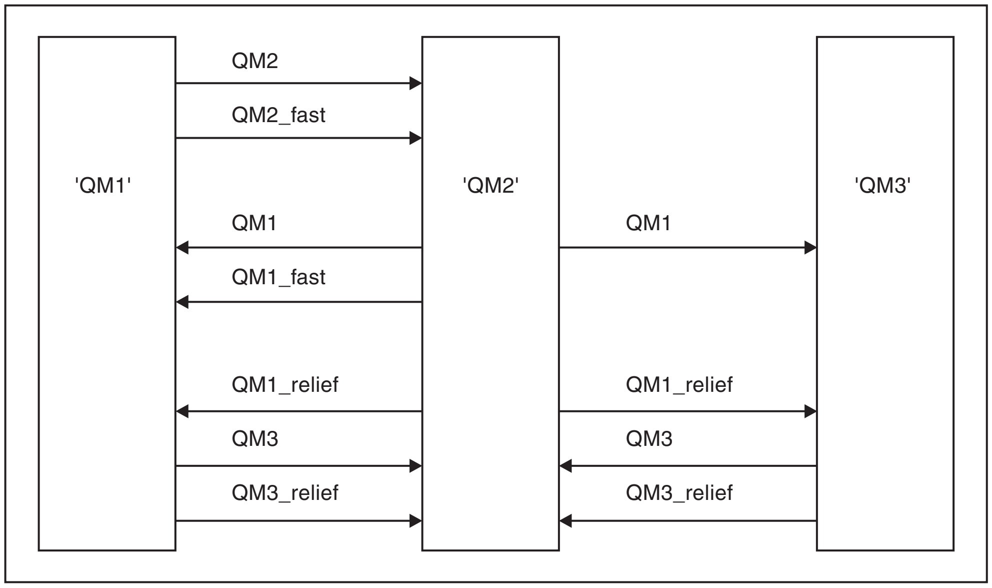


Figure 1. Network diagram showing all channels

Table 1. Example of channel names

Route name	Queue managers hosting channel	Transmission queue name	Suggested channel name
QM1	QM1 & QM2	QM1 (at QM2)	QM1.from.QM2
QM1	QM2 & QM3	QM1 (at QM3)	QM1.from.QM3
QM1_fast	QM1 & QM2	QM1_fast (at QM2)	QM1_fast.from.QM2
QM1_relief	QM1 & QM2	QM1_relief (at QM2)	QM1_relief.from.QM2
QM1_relief	QM2 & QM3	QM1_relief (at QM3)	QM1_relief.from.QM3
QM2	QM1 & QM2	QM2 (at QM1)	QM2.from.QM1
QM2_fast	QM1 & QM2	QM2_fast (at QM1)	QM2_fast.from.QM1
QM3	QM1 & QM2	QM3 (at QM1)	QM3.from.QM1
QM3	QM2 & QM3	QM3 (at QM2)	QM3.from.QM2
QM3_relief	QM1 & QM2	QM3_relief (at QM1)	QM3_relief.from.QM1
QM3_relief	QM2 & QM3	QM3_relief (at QM2)	QM3_relief.from.QM2

Note:

1. z/OS On IBM MQ for z/OS, queue manager names are limited to four characters.
2. Name all the channels in your network uniquely. As shown in Table 1, including the source and target queue manager names in the channel name is a good way to do so.

Network planner

Creating a network assumes that there is another, higher level function of *network planner* whose plans are implemented by the other members of the team.

For widely used applications, it is more economical to think in terms of local access sites for the concentration of message traffic, using wide-band links between the local access sites, as shown in Figure 2 on page 10.

In this example there are two main systems and a number of satellite systems. Tthe actual configuration would depend on business considerations. There are two concentrator queue managers located at convenient centers. Each QM-concentrator has message channels to the local queue managers:

- QM-concentrator 1 has message channels to each of the three local queue managers, QM1, QM2, and QM3. The applications using these queue managers can communicate with each other through the QM-concentrators.
- QM-concentrator 2 has message channels to each of the three local queue managers, QM4, QM5, and QM6. The applications using these queue managers can communicate with each other through the QM-concentrators.
- The QM-concentrators have message channels between themselves thus allowing any application at a queue manager to exchange messages with any other application at another queue manager.

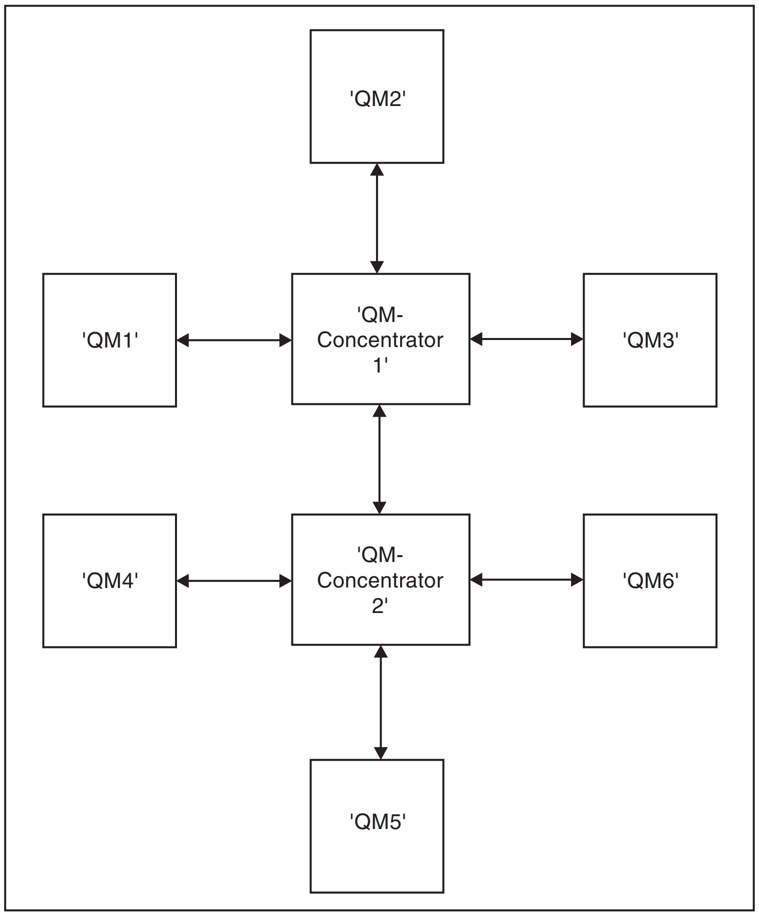


Figure 2. Network diagram showing QM-concentrators

Designing clusters

Clusters provide a mechanism for interconnecting queue managers in a way that simplifies both the initial configuration and the ongoing management. Clusters must be carefully designed, to ensure that they function correctly, and that they achieve the required levels of availability and responsiveness.

Before you begin

For an introduction to clustering concepts, see the following topics:
- Distributed queuing and clusters
- "Comparison of clustering and distributed queuing" on page 17
- Components of a cluster

When you are designing your queue manager cluster you have to make some decisions. You must first decide which queue managers in the cluster are to hold the full repositories of cluster information. Any queue manager you create can work in a cluster. You can choose any number of queue managers for this purpose but the ideal number is two. For information about selecting queue managers to hold the full repositories, see"How to choose cluster queue managers to hold full repositories" on page 19.

See the following topics for more information about designing your cluster:
- "Example clusters" on page 27
- "Organizing a cluster" on page 21
- "Cluster naming conventions" on page 22
- "Queue-sharing groups and clusters" on page 23
- "Overlapping clusters" on page 23

What to do next

See the following topics for more information about configuring and working with clusters:
- Establishing communication in a cluster
- Configuring a queue manager cluster
- Routing messages to and from clusters
- Using clusters for workload management

For more information to help you configure your cluster, see"Clustering tips" on page 24.

Planning how you use multiple cluster transmission queues:

You can explicitly define transmission queues, or have the system generate the transmission queues for you. If you define the transmission queues yourself, you have more control over the queue definitions. z/OS On z/OS, you also have more control over the page set where the messages are held.

Defining the transmission queues

There are two methods of defining transmission queues:
- Automatically, using the queue manager attribute DEFCLXQ, as follows:
  ```
  ALTER QMGR DEFCLXQ(SCTQ | CHANNEL)
  ```
  DEFCLXQ(SCTQ) indicates that the default transmission queue for all cluster-sender channels is SYSTEM.CLUSTER.TRANSMIT.QUEUE. This is the default value.

  DEFCLXQ(CHANNEL) indicates that by default each cluster-sender channel uses a separate transmission queue named SYSTEM.CLUSTER.TRANSMIT.*channel name*. Each transmission queue is automatically defined by the queue manager. See "Automatically-defined cluster transmission queues" on page 13 for more information.

- Manually, by defining a transmission queue with a value specified for the CLCHNAME attribute. The CLCHNAME attribute indicates which cluster-sender channels should use the transmission queue. See “Planning for manually-defined cluster transmission queues” on page 15for more information.

What security do I need?

To initiate a switch, either automatically or manually, you need authority to start a channel.

To define the queue used as a transmission queue, you need standard IBM MQ authority to define the queue.

When is a suitable time to implement the change?

When changing the transmission queue used by cluster-sender channels, you need to allocate a time in which to make the update, considering the following points:

- The time required for a channel to switch transmission queue depends on the total number of messages on the old transmission queue, how many messages need to be moved, and the size of the messages.
- Applications can continue to put messages to the transmission queue while the change is happening. This might lead to an increase in the transition time.
- You can change the CLCHNAME parameter of any transmission queue or DEFCLXQ at any time, preferably when the workload is low.

  Note that nothing happens immediately.
- Changes occur only when a channel starts or restarts. When a channel starts it checks the current configuration and switches to a new transmission queue if required.
- There are several changes that might alter the association of a cluster-sender channel with a transmission queue:
  - Altering the value of a transmission queue's CLCHNAME attribute, making CLCHNAME less specific or blank.
  - Altering the value of a transmission queue's CLCHNAME attribute, making CLCHNAME more specific.
  - Deleting a queue with CLCHNAME specified.
  - Altering the queue manager attribute DEFCLXQ.

How long will the switch take?

During the transition period, any messages for the channel are moved from one transmission queue to another. The time required for a channel to switch transmission queue depends on the total number of messages on the old transmission queue, and how many messages need to be moved.

For queues containing a few thousand messages, it should take under a second to move the messages. The actual time depends on the number and size of the messages. Your queue manager should be able to move messages at many megabytes each second.

Applications can continue to put messages to the transmission queue while the change is happening. This might lead to an increase in the transition time.

Each affected cluster-sender channel must be restarted for the change to take effect. Therefore, it is best to change the transmission queue configuration when the queue manager is not busy, and few messages are stored on the cluster transmission queues.

The **`runswchl`** command, z/OS or the SWITCH CHANNEL(*) STATUS command in CSQUTIL on z/OS, can be used to query the status of cluster-sender channels and what pending changes are outstanding to their transmission queue configuration.

How to implement the change

See Implementing the system using multiple cluster transmission queues for details on how you make the change to multiple cluster transmission queues, either automatically or manually.

Undoing the change

See Undoing a change for details on how you back out changes if you encounter problems.

Automatically-defined cluster transmission queues:

You can have the system generate the transmission queues for you.

About this task

If a channel does not have a manually defined cluster transmission queue that is associated with it, and you specify DEFCLXQ(CHANNEL), when the channel starts the queue manager automatically defines a permanent-dynamic queue for the cluster sender channel. This channel has the name `SYSTEM.CLUSTER.TRANSMIT.ChannelName`, and uses the model queue `SYSTEM.CLUSTER.TRANSMIT.MODEL`.

z/OS To set up the cluster transmission queues manually, see "Planning for manually-defined cluster transmission queues" on page 15.

Important: z/OS

If the queue manager is migrated to IBM MQ Version 8.0, the queue manager does not have the SYSTEM.CLUSTER.TRANSMIT.MODEL.QUEUE.

Define this queue first, so that the command ALTER QGMR DEFCLXQ(CHANNEL) takes effect.

The following JCL is an example of the code you can use to define the model queue:

```
//CLUSMODL JOB MSGCLASS=H,NOTIFY=&SYSUID
/*JOBPARM SYSAFF=(MVCC)
//MQCMD EXEC PGM=CSQUTIL,REGION=4096K,PARM='CDLK'
//STEPLIB DD DISP=SHR,DSN=SCEN.MQ.V000.COM.BASE.SCSQAUTH
// DD DISP=SHR,DSN=SCEN.MQ.V000.COM.BASE.SCSQANLE
//SYSPRINT DD SYSOUT=*
//SYSIN DD *
COMMAND DDNAME(CMDINP)
/*
//CMDINP DD *
DEFINE QMODEL( 'SYSTEM.CLUSTER.TRANSMIT.MODEL.QUEUE' ) +
QSGDISP( QMGR ) +

* COMMON QUEUE ATTRIBUTES
DESCR( 'SYSTEM CLUSTERING TRANSMISSION MODEL QUEUE' ) +
PUT( ENABLED ) +
DEFPRTY( 5 ) +
DEFPSIST( YES ) +

* MODEL QUEUE ATTRIBUTES
DEFTYPE( PERMDYN ) +

* LOCAL QUEUE ATTRIBUTES
GET( ENABLED ) +
SHARE +
DEFSOPT( EXCL ) +
MSGDLVSQ( PRIORITY ) +
RETINTVL( 999999999 ) +
MAXDEPTH( 999999999 ) +
MAXMSGL( 4194304 ) +
NOHARDENBO +
BOTHRESH( 0 ) +
BOQNAME( ' ' ) +
STGCLASS( 'REMOTE' ) +
USAGE( XMITQ ) +
INDXTYPE( CORRELID ) +
CFSTRUCT( ' ' ) +
MONQ( OFF ) ACCTQ( OFF ) +

* EVENT CONTROL ATTRIBUTES
QDPMAXEV( ENABLED ) +
QDPHIEV( DISABLED ) +
QDEPTHHI( 80 ) +
QDPLOEV( DISABLED ) +
QDEPTHLO( 40 ) +
QSVCIEV( NONE ) +
QSVCINT( 999999999 ) +

* TRIGGER ATTRIBUTES
TRIGGER +
TRIGTYPE( FIRST ) +
TRIGMPRI( 0 ) +
TRIGDPTH( 1 ) +
TRIGDATA( ' ' ) +
PROCESS( ' ' ) +
INITQ( ' ' )
/*
```

Procedure

1. Use the *DEFCLXQ* queue manager attribute. For more information on this attribute, see ALTER QMGR.

   There are two options:

   SCTQ This option is the default, and means that you use the single SYSTEM.CLUSTER.TRANSMIT.QUEUE.

   CHANNEL
   Means that you use multiple cluster transmission queues.
2. To switch to the new association:
   - Stop and restart the channel.
   - The channel uses the new transmission queue definition.
   - Messages are transferred by a transitional switch process from the old queue to the new transmission queue.

   Note that any application messages are put to the old definition.

   When the number of messages on the old queue reaches zero, new messages are placed directly on the new transmission queue.
3. To monitor when the switching process finishes:
   a. A switch of transmission queue that is initiated by a channel runs in the background and your administrator can monitor the queue manager job log to determine when it has completed.
   b. Monitor messages on the job log to show the progress of the switch.
   c. To make sure that only the channels that you wanted are using this transmission queue, issue the command DIS CLUSQMGR(*) where, for example, the transmission queue property that defines the transmission queue is `APPQMGR.CLUSTER1.XMITQ`.
   d. z/OS Use the SWITCH CHANNEL (*) STATUS command under CSQUTIL. This option tells you what pending changes are outstanding, and how many messages need to be moved between transmission queues.

Results

You have set up your cluster transmission queue, or queues.

Related tasks:

"Planning for manually-defined cluster transmission queues"
If you define the transmission queues yourself you have more control over the definitions, and the page set on which the messages are held.

Related information:

ALTER QMGR

DISPLAY CLUSQMGR

Planning for manually-defined cluster transmission queues:

If you define the transmission queues yourself you have more control over the definitions, and the page set on which the messages are held.

About this task

Your administrator manually defines a transmission queue and uses a new queue attribute CLCHNAME to define which cluster sender channel, or channels, will use this queue as their transmission queue.

Note that CLCHNAME can include a wildcard character at the beginning, or end, to allow a single queue to be used for multiple channels.

To set up cluster transmission queues automatically, see "Automatically-defined cluster transmission queues" on page 13.

Procedure

1. For example, enter the following:
   ```
   DEFINE QLOCAL(APPQMGR.CLUSTER1.XMITQ)
   CLCHNAME(CLUSTER1.TO.APPQMGR)
   USAGE(XMITQ) STGCLASS(STG1)
   INDXTYPE( CORRELID ) SHARE

   DEFINE STGCLASS(STG1) PSID(3)
   DEFINE PSID(3) BUFFERPOOL(4)
   ```

   Tip: You need to plan which page set (and buffer pool) you use for your transmission queues. You can have different page sets for different queues, and provide isolation between them, so one page set filling up, does not impact transmission queues in other page sets.

   See Working with cluster transmission queues and cluster-sender channels for information on how each channel selects the appropriate queue.

   When the channel starts it switches its association to the new transmission queue. In order to make sure no message is lost, the queue manager automatically transfers messages from the old cluster transmission queue to the new transmission queue in order.
2. Use the CSQUTIL SWITCH function to change to the new association. See Switch the transmission queue associated with cluster-sender channels (SWITCH) for further information.
   a. STOP the channel, or channels, whose transmission queue is to be changed, so that they are in STOPPED status. For example:
      ```
      STOP CHANNEL(CLUSTER1.TO.APPQMGR)
      ```
   b. Change the `CLCHNAME(XXXX)` attribute on the transmission queue.
   c. Use the SWITCH function to switch the messages or monitor what is happening. Use the command
      ```
      SWITCH CHANNEL(*) MOVEMSGS(YES)
      ```

      to move the messages without starting the channel.

d. Start the channel, or channels, and check whether the channel is using the correct queues. For example:

```
DIS CHS(CLUSTER1.TO.APPQMGR)
DIS CHS(*) where(XMITQ eq APPQMGR.CLUSTER1.XMITQ)
```

Tip:

- The following process uses the CSQUTIL SWITCH function; for more information, see Switch the transmission queue associated with cluster-sender channels (SWITCH).

  You do not have to use this function but using this function gives more options:

  – Using SWITCH CHANNEL (*) STATUS provides an easy way to identify the switching status of cluster-sender channels. It allows your administrator to see what channels are currently switching, and those channels having a switch pending that take effect when those channels next start.

    Without this capability your administrator needs to use multiple DISPLAY commands, and then process the resulting output to ascertain this information. Your administrator can also confirm that a configuration change has the required result.

  – If CSQUTIL is used to initiate the switch, CSQUTIL continues to monitor the progress of this operation, and only ends when the switch has completed.

    This can make it much easier to perform these operations in batch. Also, if CSQUTIL is run to switch multiple channels, CSQUTIL performs these actions sequentially; this can have less impact for your enterprise than multiple switches running in parallel.

Results

You have set up your cluster transmission queue, or queues.

Access control and multiple cluster transmission queues:

Choose between three modes of checking when an application puts messages to remote cluster queues. The modes are checking remotely against the cluster queue, checking locally against `SYSTEM.CLUSTER.TRANSMIT.QUEUE`, or checking against local profiles for the cluster queue, or cluster queue manager.

IBM MQ gives you the choice of checking locally, or locally and remotely, that a user has permission to put a message to a remote queue. A typical IBM MQ application uses local checking only, and relies on the remote queue manager trusting the access checks made on the local queue manager. If remote checking is not used, the message is put to the target queue with the authority of the remote message channel process. To use remote checking you must set the put authority of the receiving channel to context security.

The local checks are made against the queue that the application opens. In distributed queuing, the application usually opens a remote queue definition, and access checks are made against the remote queue definition. If the message is put with a full routing header, the checks are made against the transmission queue. If an application opens a cluster queue that is not on the local queue manager, there is no local object to check. The access control checks are made against the cluster transmission queue, `SYSTEM.CLUSTER.TRANSMIT.QUEUE`. Even with multiple cluster transmission queues, from Version 7.5, local access control checks for remote cluster queues are made against `SYSTEM.CLUSTER.TRANSMIT.QUEUE`.

The choice of local or remote checking is a choice between two extremes. Checking remotely is fine-grained. Every user must have an access control profile on every queue manager in the cluster to put to any cluster queue. Checking locally is coarse-grained. Every user needs only one access control profile for the cluster transmission queue on the queue manager they are connected to. With that profile, they can put a message to any cluster queue on any queue manager in any cluster.

Since Version 7.1, administrators have another way to set up access control for cluster queues. You can create a security profile for a cluster queue on any queue manager in the cluster using the **setmqaut** command. The profile takes affect if you open a remote cluster queue locally, specifying only the queue name. You can also set up a profile for a remote queue manager. If you do so, the queue manager can check the profile of a user that opens a cluster queue by providing a fully qualified name.

The new profiles work only if you change the queue manager stanza, **ClusterQueueAccessControl** to RQMName. The default is Xmitq. You must create profiles for all the cluster queues existing applications use cluster queues. If you change the stanza to RQMName without creating profiles the applications are likely to fail.

Tip: The changes made to cluster queue accessing checking in Version 7.1 do not apply to remote queuing. Access checks are still made against local definitions. The changes mean that you can follow the same approach to configure access checking on cluster queues and cluster topics. z/OS The changes also align the access checking approach for cluster queues more closely with z/OS. The commands to set up access checking on z/OS are different, but both check access against a profile rather than against the object itself.

Related information:

Clustering: Application isolation using multiple cluster transmission queues

Setting ClusterQueueAccessControl

Comparison of clustering and distributed queuing:

Compare the components that need to be defined to connect queue managers using distributed queuing and clustering.

If you do not use clusters, your queue managers are independent and communicate using distributed queuing. If one queue manager needs to send messages to another, you must define:

- A transmission queue
- A channel to the remote queue manager

Figure 3 shows the components required for distributed queuing.

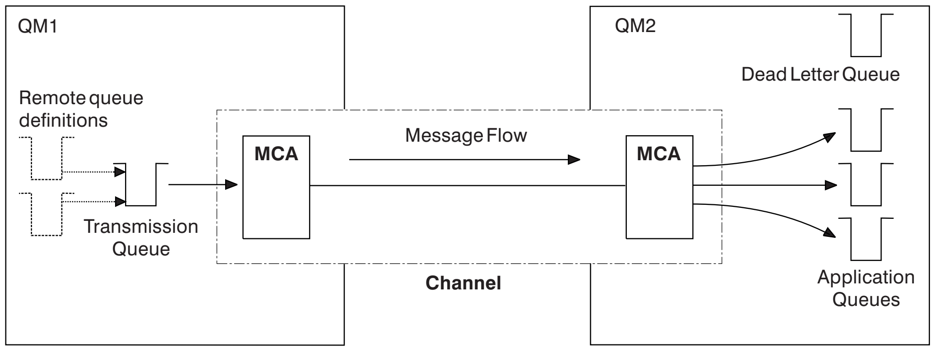


Figure 3. Distributed queuing

If you group queue managers in a cluster, queues on any queue manager are available to any other queue manager in the cluster. Any queue manager can send a message to any other queue manager in the same cluster without explicit definitions. You do not provide channel definitions, remote-queue definitions, or transmission queues for each destination. Every queue manager in a cluster has a single transmission queue from which it can transmit messages to any other queue manager in the cluster. Each queue manager in a cluster needs to define only:

- One cluster-receiver channel on which to receive messages

- One cluster-sender channel with which it introduces itself and learns about the cluster

Definitions to set up a cluster versus distributed queuing

Look at Figure 4, which shows four queue managers each with two queues. Consider how many definitions are needed to connect these queue managers using distributed queuing. Compare how many definitions are needed to set up the same network as a cluster.

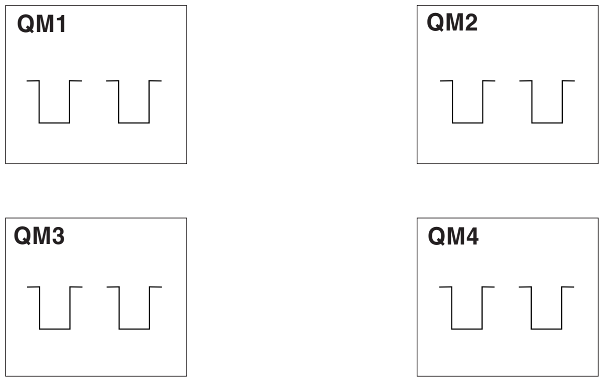


Figure 4. A network of four queue managers

Definitions to set up a network using distributed queuing

To set up the network shown in Figure 3 on page 17 using distributed queuing, you might have the following definitions:

Table 2. Definitions for distributed queuing

Description	Number per queue manager	Total number
A sender-channel definition for a channel on which to send messages to every other queue manager	3	12
A receiver-channel definition for a channel on which to receive messages from every other queue manager	3	12
A transmission-queue definition for a transmission queue to every other queue manager	3	12
A local-queue definition for each local queue	2	8
A remote-queue definition for each remote queue to which this queue manager wants to put messages	6	24

You might reduce this number of definitions by using generic receiver-channel definitions. The maximum number of definitions could be as many as 17 on each queue manager, which is a total of 68 for this network.

Definitions to set up a network using clusters

To set up the network shown in Figure 3 on page 17 using clusters you need the following definitions:

Table 3. Definitions for clustering

Description	Number per queue manager	Total number
A cluster-sender channel definition for a channel on which to send messages to a repository queue manager	1	4
A cluster-receiver channel definition for a channel on which to receive messages from other queue managers in the cluster	1	4
A local-queue definition for each local queue	2	8

To set up this cluster of queue managers (with two full repositories), you need four definitions on each queue manager, a total of sixteen definitions altogether. You also need to alter the queue manager definitions for two of the queue managers, to make them full repository queue managers for the cluster.

Only one CLUSSDR and one CLUSRCVR channel definition is required. When the cluster is defined, you can add or remove queue managers (other than the repository queue managers) without any disruption to the other queue managers.

Using a cluster reduces the number of definitions required to set up a network containing many queue managers.

With fewer definitions to make there is less risk of error:
- Object names always match, for example the channel name in a sender-receiver pair.
- The transmission queue name specified in a channel definition always matches the correct transmission queue definition or the transmission queue name specified in a remote queue definition.
- A QREMOTE definition always points to the correct queue at the remote queue manager.

Once a cluster is set up, you can move cluster queues from one queue manager to another within the cluster without having to do any system management work on any other queue manager. There is no chance of forgetting to delete or modify channel, remote-queue, or transmission-queue definitions. You can add new queue managers to a cluster without any disruption to the existing network.

How to choose cluster queue managers to hold full repositories:

In each cluster you must choose at least one, and preferably two queue managers to hold full repositories. Two full repositories are sufficient for all but the most exceptional circumstances. If possible, choose queue managers that are hosted on robust and permanently-connected platforms, that do not have coinciding outages, and that are in a central position geographically. Also consider dedicating systems as full repository hosts, and not using these systems for any other tasks.

Full repositories are queue managers that maintain a complete picture of the state of the cluster. To share this information, each full repository is connected by CLUSSDR channels (and their corresponding CLUSRCVR definitions) to every other full repository in the cluster. You must manually define these channels.

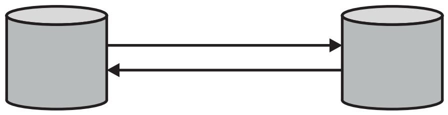


Figure 5. Two connected full repositories.

Every other queue manager in the cluster maintains a picture of what it currently knows about the state of the cluster in a *partial repository*. These queue managers publish information about themselves, and request information about other queue managers, using any two available full repositories. If a chosen full repository is not available, another is used. When the chosen full repository becomes available again, it collects the latest new and changed information from the others so that they keep in step. If all the full repositories go out of service, the other queue managers use the information they have in their partial repositories. However, they are limited to using the information that they have; new information and requests for updates cannot be processed. When the full repositories reconnect to the network, messages are exchanged to bring all repositories (both full and partial) up to date.

When planning the allocation of full repositories, include the following considerations:

- The queue managers chosen to hold full repositories need to be reliable and managed. Choose queue managers that are hosted on a robust and permanently-connected platform.
- Consider the planned outages for the systems hosting your full repositories, and ensure that they do not have coinciding outages.
- Consider network performance: Choose queue managers that are in a central position geographically, or that share the same system as other queue managers in the cluster.
- Consider whether a queue manager is a member of more than one cluster. It can be administratively convenient to use the same queue manager to host the full repositories for several clusters, provided this benefit is balanced against how busy you expect the queue manager to be.
- Consider dedicating some systems to contain only full repositories, and not using these systems for any other tasks. This ensures that these systems only require maintenance for queue manager configuration, and are not removed from service for the maintenance of other business applications. It also ensures that the task of maintaining the repository does not compete with applications for system resources. This can be particularly beneficial in large clusters (say, clusters of more than a thousand queue managers), where full repositories have a much higher workload in maintaining the cluster state.

Having more than two full repositories is possible, but rarely recommended. Although object definitions (that is, queues, topics and channels) flow to all available full repositories, requests only flow from a partial repository to a maximum of two full repositories. This means that, when more than two full repositories are defined, and any two full repositories become unavailable, some partial repositories might not receive updates they would expect. See WMQ Clusters: Why only two Full Repositories?

One situation in which you might find it useful to define more than two full repositories is when migrating existing full repositories to new hardware or new queue managers. In this case, you should introduce the replacement full repositories, and confirm that they have become fully populated, before you remove the previous full repositories. Whenever you add a full repository, remember that you must directly connect it to every other full repository with `CLUSSDR` channels.

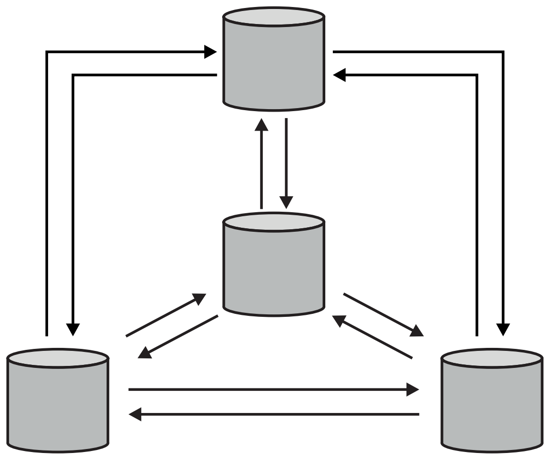


Figure 6. More than two connected full repositories

Related information:

WMQ Clusters: Why only two Full Repositories?

How big can a WMQ Cluster be?

Organizing a cluster:

Select which queue managers to link to which full repository. Consider the performance effect, the queue manager version, and whether multiple CLUSSDR channels are desirable.

Having selected the queue managers to hold full repositories, you need to decide which queue managers to link to which full repository. The CLUSSDR channel definition links a queue manager to a full repository from which it finds out about the other full repositories in the cluster. From then on, the queue manager sends messages to any two full repositories. It always tries to use the one to which it has a CLUSSDR channel definition first. You can choose to link a queue manager to either full repository. In choosing, consider the topology of your configuration, and the physical or geographical location of the queue managers.

z/OS Queue managers on IBM MQ z/OS Version 6.0 or later support PCF commands. The IBM MQ Explorer can use them to query information about the cluster (in just the same way as with distributed queue managers).

z/OS z/OS queue managers up to IBM MQ v5.3.1 do not support PCF commands. The IBM MQ Explorer cannot use them to query information about the cluster. For this reason, the IBM MQ Explorer does not allow you to select IBM MQ v5.3.1 z/OS queue managers at the source of the information about a cluster.

Because all cluster information is sent to two full repositories, there might be situations in which you want to make a second CLUSSDR channel definition. You might define a second CLUSSDR channel in a cluster that has many full repositories spread over a wide area. You can then control which two full repositories your information is sent to.

Cluster naming conventions:

Consider naming queue managers in the same cluster using a naming convention that identifies the cluster to which the queue manager belongs. Use a similar naming convention for channel names and extend it to describe the channel characteristics. z/OS Do not use a generic connection in defining cluster-receiver channels on z/OS.

This information contains the old guidance on naming conventions, and the current guidance. As the IBM MQ technology improves, and as customers use technology in new or different ways, new recommendations and information must be provided for these scenarios.

Cluster naming conventions: Old best practices

When setting up a new cluster, consider a naming convention for the queue managers. Every queue manager must have a different name. If you give queue managers in a cluster a set of similar names, it might help you to remember which queue managers are grouped where.

When defining channels, remember that all cluster-sender channels have the same name as their corresponding cluster-receiver channel. Channel names are limited to a maximum of 20 characters.

Every cluster-receiver channel must also have a unique name. One possibility is to use the queue manager name preceded by the cluster name. For example, if the cluster name is `CLUSTER1` and the queue managers are `QM1`, `QM2`, then cluster-receiver channels are `CLUSTER1.QM1`, `CLUSTER1.QM2`.

You might extend this convention if channels have different priorities or use different protocols; for example, `CLUSTER1.QM1.S1`, `CLUSTER1.QM1.N3`, and `CLUSTER1.QM1.T4`. In this example, `S1` might be the first SNA channel, `N3` might be the NetBIOS channel with a network priority of three.

A final qualifier might describe the class of service the channel provides.

z/OS In IBM MQ for z/OS, you can define VTAM generic resources or *Dynamic Domain Name Server* (DDNS) generic names. You can define connection names using generic names. However, when you create a cluster-receiver definition, do not use a generic connection name.

The problem with using generic connection names for cluster-receiver definitions is as follows. If you define a `CLUSRCVR` with a generic `CONNAME` there is no guarantee that your `CLUSSDR` channels point to the queue managers you intend. Your initial `CLUSSDR` might end up pointing to any queue manager in the queue-sharing group, not necessarily one that hosts a full repository. If a channel starts trying a connection again, it might reconnect to a different queue manager with the same generic name disrupting the flow of messages.

Cluster naming conventions: Current best practices

A good naming convention can help to reduce confusion over ownership and scope of clusters. A clear naming convention throughout the cluster topology causes a lot less confusion if clusters get merged at a later time. This situation is also improved if everyone involved is clear about who owns which queue managers and which clusters. Probably the most important point for cluster naming conventions is to put the queue manager name in the channel name, see the following example:

```
CLUSNAME.QMGRNAME
```

This convention might not be obvious to experienced IBM MQ users that are unfamiliar with clusters. This oversight is because the `XXX.TO.YYY` format is such a common method. For example, `CLUSTER.TO.XX` or `CLUSTER.X` are commonly used formats that are not recommended for clustering, as they can quickly reach the 20 character limit. The commonly used `CLUSTER.TO.XX` format becomes confusing if another channel is added later (for example when joining another cluster).

Other objects also benefit from sensible rules, such as: `LOB.PROJECT.QNAME` or `LOB.CLUSTER.ALIAS.NAME`.

Queue-sharing groups and clusters:

Shared queues can be cluster queues and queue managers in a queue-sharing group can also be cluster queue managers.

z/OS On IBM MQ for z/OS you can group queue managers into queue-sharing groups. A queue manager in a queue-sharing group can define a local queue that is to be shared by up to 32 queue managers.

Shared queues can also be cluster queues. Furthermore, the queue managers in a queue-sharing group can also be in one or more clusters.

z/OS In IBM MQ for z/OS, you can define VTAM generic resources or *Dynamic Domain Name Server* (DDNS) generic names. You can define connection names using generic names. However, when you create a cluster-receiver definition, do not use a generic connection name.

z/OS The problem with using generic connection names for cluster-receiver definitions is as follows. If you define a `CLUSRCVR` with a generic `CONNAME` there is no guarantee that your `CLUSSDR` channels point to the queue managers you intend. Your initial `CLUSSDR` might end up pointing to any queue manager in the queue-sharing group, not necessarily one that hosts a full repository. If a channel starts trying a connection again, it might reconnect to a different queue manager with the same generic name disrupting the flow of messages.

Overlapping clusters:

Overlapping clusters provide additional administrative capabilities. Use namelists to reduce the number of commands needed to administer overlapping clusters.

You can create clusters that overlap. There are a number of reasons you might define overlapping clusters; for example:
- To allow different organizations to have their own administration.
- To allow independent applications to be administered separately.
- To create classes of service.

In Figure 7 on page 24, the queue manager `STF2` is a member of both the clusters. When a queue manager is a member of more than one cluster, you can take advantage of namelists to reduce the number of definitions you need. Namelists contain a list of names, for example, cluster names. You can create a namelist naming the clusters. Specify the namelist on the `ALTER QMGR` command for `STF2` to make it a full repository queue manager for both clusters.

If you have more than one cluster in your network, you must give them different names. If two clusters with the same name are ever merged, it is not possible to separate them again. It is also a good idea to give the clusters and channels different names. They are more easily distinguished when you look at the output from the `DISPLAY` commands. Queue manager names must be unique within a cluster for it to work correctly.

Defining classes of service

Imagine a university that has a queue manager for each member of staff and each student. Messages between members of staff are to travel on channels with a high priority and high bandwidth. Messages between students are to travel on cheaper, slower channels. You can set up this network using traditional distributed queuing techniques. IBM MQ selects which channels to use by looking at the destination queue name and queue manager name.

To clearly differentiate between the staff and students, you could group their queue managers into two clusters as shown in Figure 7. IBM MQ moves messages to the meetings queue in the staff cluster only over channels that are defined in that cluster. Messages for the gossip queue in the students cluster go over channels defined in that cluster and receive the appropriate class of service.

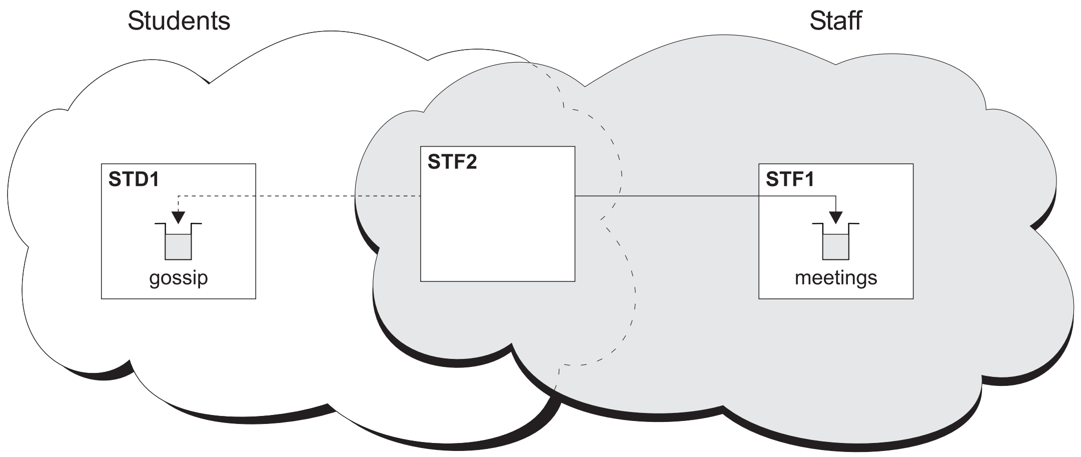


Figure 7. Classes of service

Clustering tips:

You might need to make some changes to your systems or applications before using clustering. There are both similarities and differences from the behavior of distributed queuing.

- z/OS The IBM MQ Explorer cannot directly administer IBM MQ for z/OS queue managers at versions earlier than Version 6.0.
- You must add manual configuration definitions to queue managers outside a cluster for them to access cluster queues.
- If you merge two clusters with the same name, you cannot separate them again. Therefore it is advisable to give all clusters a unique name.
- If a message arrives at a queue manager but there is no queue there to receive it, the message is put on the dead-letter queue. If there is no dead-letter queue, the channel fails and tries again. The use of the dead-letter queue is the same as with distributed queuing.
- The integrity of persistent messages is maintained. Messages are not duplicated or lost as a result of using clusters.
- Using clusters reduces system administration. Clusters make it easy to connect larger networks with many more queue managers than you would be able to contemplate using distributed queuing. There is a risk that you might consume excessive network resources if you attempt to enable communication between every queue manager in a cluster.
- If you use the IBM MQ Explorer, which presents the queue managers in a tree structure, the view for large clusters might be cumbersome.
- z/OS IBM MQ Explorer can administer a cluster with repository queue managers on IBM MQ for z/OS Version 6 or later. You need not nominate an additional repository on a separate system. For earlier versions of IBM MQ on z/OS, the IBM MQ Explorer cannot administer a cluster with repository queue managers. You must nominate an additional repository on a system that the IBM MQ Explorer can administer.

- Multi The purpose of distribution lists is to use a single MQPUT command to send the same message to multiple destinations. Distribution lists are supported on IBM MQ for Multiplatforms. You can use distribution lists with queue manager clusters. In a cluster, all the messages are expanded at MQPUT time. The advantage, in terms of network traffic, is not so great as in a non-clustering environment. The advantage of distribution lists is that the numerous channels and transmission queues do not need to be defined manually.
- If you are going to use clusters to balance your workload examine your applications. See whether they require messages to be processed by a particular queue manager or in a particular sequence. Such applications are said to have message affinities. You might need to modify your applications before you can use them in complex clusters.
- You might choose to use the `MQOO_BIND_ON_OPEN` option on an MQOPEN to force messages to be sent to a specific destination. If the destination queue manager is not available the messages are not delivered until the queue manager becomes available again. Messages are not routed to another queue manager because of the risk of duplication.
- If a queue manager is to host a cluster repository, you need to know its host name or IP address. You have to specify this information in the `CONNAME` parameter when you make the `CLUSSDR` definition on other queue managers joining the cluster. If you use DHCP, the IP address is subject to change because DHCP can allocate a new IP address each time you restart a system. Therefore, you must not specify the IP address in the `CLUSSDR` definitions. Even if all the `CLUSSDR` definitions specify the host name rather than the IP address, the definitions would still not be reliable. DHCP does not necessarily update the DNS directory entry for the host with the new address. If you must nominate queue managers as full repositories on systems that use DHCP, install software that guarantees to keep your DNS directory up to date.
- Do not use generic names, for example VTAM generic resources or Dynamic Domain Name Server (DDNS) generic names as the connection names for your channels. If you do, your channels might connect to a different queue manager than expected.
- You can only get a message from a local cluster queue, but you can put a message to any queue in a cluster. If you open a queue to use the MQGET command, the queue manager opens the local queue.
- You do not need to alter any of your applications if you set up a simple IBM MQ cluster. The application can name the target queue on the MQOPEN call and does not need to know about the location of the queue manager. If you set up a cluster for workload management you must review your applications and modify them as necessary.
- You can view current monitoring and status data for a channel or queue using the `DISPLAY CHSTATUS` and the `DISPLAY QSTATUS` **`runmqsc`** commands. The monitoring information can be used to help gauge the performance and health of the system. Monitoring is controlled by queue manager, queue, and channel attributes. Monitoring of auto-defined cluster-sender channels is possible with the `MONACLS` queue manager attribute.

Related concepts:

"Comparison of clustering and distributed queuing" on page 17
Compare the components that need to be defined to connect queue managers using distributed queuing and clustering.

Related information:

Clusters

Configuring a queue manager cluster

Components of a cluster

Setting up a new cluster

How long do the queue manager repositories retain information?:

Queue manager repositories retain information for 30 days. An automatic process efficiently refreshes information that is being used.

When a queue manager sends out some information about itself, the full and partial repository queue managers store the information for 30 days. Information is sent out, for example, when a queue manager advertises the creation of a new queue. To prevent this information from expiring, queue managers automatically resend all information about themselves after 27 days. If a partial repository sends a new request for information part way through the 30 day lifetime, the expiry time remains the original 30 days.

When information expires, it is not immediately removed from the repository. Instead it is held for a grace period of 60 days. If no update is received within the grace period, the information is removed. The grace period allows for the fact that a queue manager might have been temporarily out of service at the expiry date. If a queue manager becomes disconnected from a cluster for more than 90 days, it stops being part of the cluster. However, if it reconnects to the network it becomes part of the cluster again. Full repositories do not use information that has expired to satisfy new requests from other queue managers.

Similarly, when a queue manager sends a request for up-to-date information from a full repository, the request lasts for 30 days. After 27 days IBM MQ checks the request. If it has been referenced during the 27 days, it is refreshed automatically. If not, it is left to expire and is refreshed by the queue manager if it is needed again. Expiring requests prevents a buildup of requests for information from dormant queue managers.

Note: For large clusters, it can be disruptive if many queue managers automatically resend all information about themselves at the same time. See Refreshing in a large cluster can affect performance and availability of the cluster.

Related information:
Clustering: Using REFRESH CLUSTER best practices

Example clusters:

The first example shows the smallest possible cluster of two queue managers. The second and third examples show two versions of a three queue manager cluster.

The smallest possible cluster contains only two queue managers. In this case both queue managers contain full repositories. You need only a few definitions to set up the cluster, and yet there is a high degree of autonomy at each queue manager.

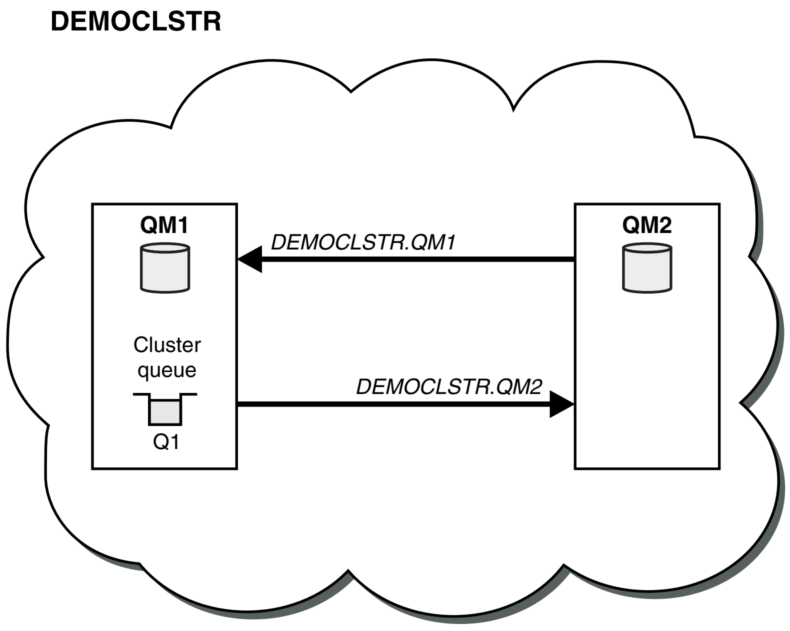


Figure 8. A small cluster of two queue managers

- The queue managers can have long names such as `LONDON` and `NEWYORK`. z/OS On IBM MQ for z/OS, queue manager names are limited to four characters.
- Each queue manager is typically configured on a separate machine. However, you can have multiple queue managers on the same machine.

For instructions on setting up a similar example cluster, see Setting up a new cluster.

Figure 9 on page 28 shows the components of a cluster called `CLSTR1`.
- In this cluster, there are three queue managers, `QM1`, `QM2`, and `QM3`.
- `QM1` and `QM2` host repositories of information about all the queue managers and cluster-related objects in the cluster. They are referred to as *full repository queue managers*. The repositories are represented in the diagram by the shaded cylinders.
- `QM2` and `QM3` host some queues that are accessible to any other queue manager in the cluster. Queues that are accessible to any other queue manager in the cluster are called *cluster queues*. The cluster queues are represented in the diagram by the shaded queues. Cluster queues are accessible from anywhere in the cluster. IBM MQ clustering code ensures that remote queue definitions for cluster queues are created on any queue manager that refers to them.

As with distributed queuing, an application uses the MQPUT call to put a message on a cluster queue at any queue manager in the cluster. An application uses the MQGET call to retrieve messages from a cluster queue only on the queue manager where the queue resides.

- Each queue manager has a manually created definition for the receiving end of a channel called *cluster_name*. *queue_manager_name* on which it can receive messages. On the receiving queue manager, *cluster_name*. *queue_manager_name* is a cluster-receiver channel. A cluster-receiver channel is like a receiver channel used in distributed queuing; it receives messages for the queue manager. In addition, it also receives information about the cluster.
-

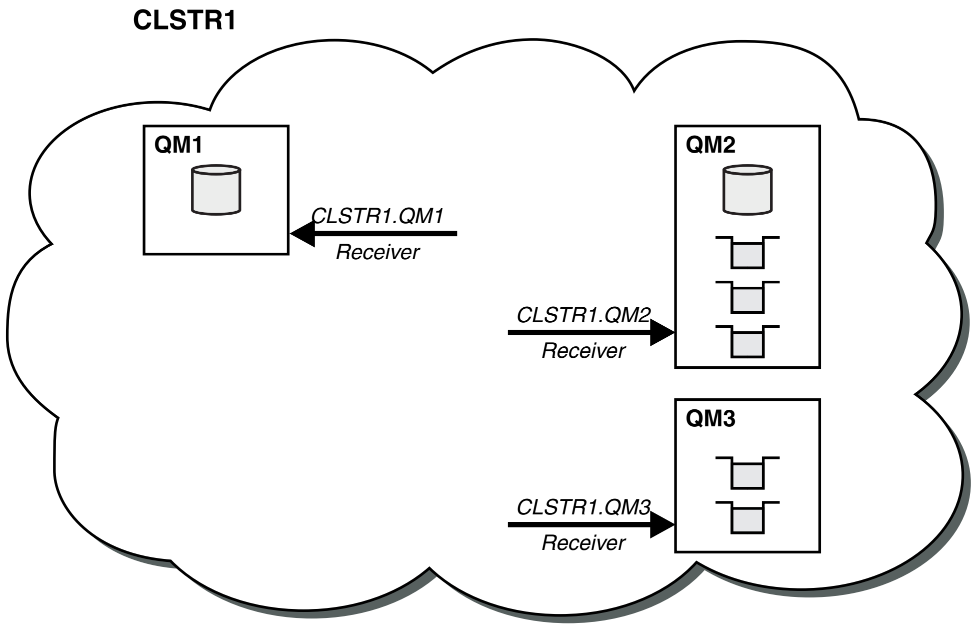


Figure 9. A cluster of queue managers

- In Figure 10 on page 29 each queue manager also has a definition for the sending end of a channel. It connects to the cluster-receiver channel of one of the full repository queue managers. On the sending queue manager, *cluster_name*. *queue_manager_name* is a cluster-sender channel. QM1 and QM3 have cluster-sender channels connecting to CLSTR1.QM2, see dotted line "2".

  QM2 has a cluster-sender channel connecting to CLSTR1.QM1, see dotted line "3". A cluster-sender channel is like a sender-channel used in distributed queuing; it sends messages to the receiving queue manager. In addition, it also sends information about the cluster.

  Once both the cluster-receiver end and the cluster-sender end of a channel are defined, the channel starts automatically.

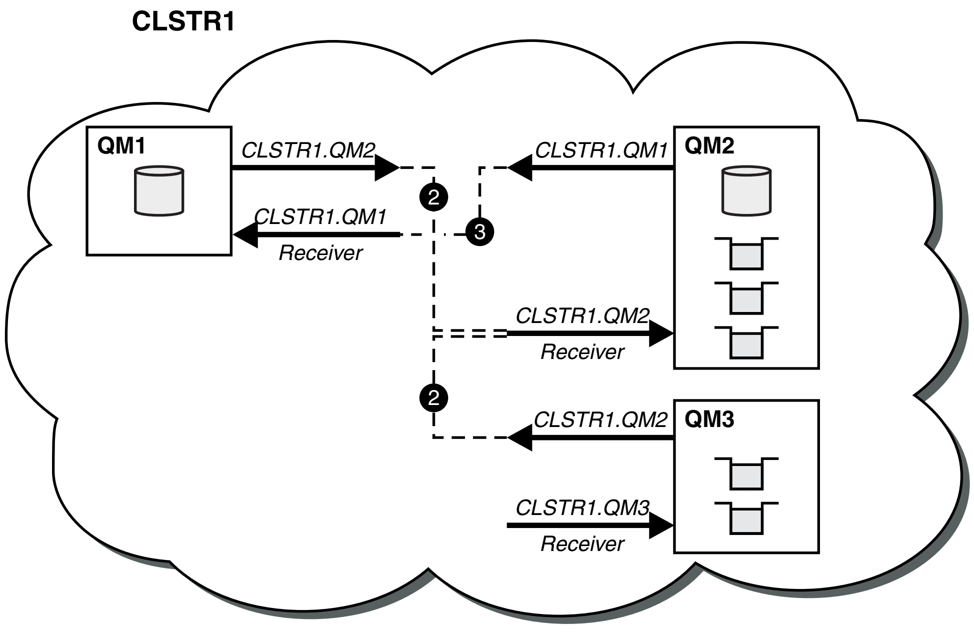


Figure 10. A cluster of queue managers with sender channels

Defining a cluster-sender channel on the local queue manager introduces that queue manager to one of the full repository queue managers. The full repository queue manager updates the information in its full repository accordingly. Then it automatically creates a cluster-sender channel back to the original queue manager, and sends that queue manager information about the cluster. Thus a queue manager learns about a cluster and a cluster learns about a queue manager.

Look again at Figure 9 on page 28. Suppose that an application connected to queue manager `QM3` wants to send some messages to the queues at `QM2`. The first time that `QM3` must access those queues, it discovers them by consulting a full repository. The full repository in this case is `QM2`, which is accessed using the sender channel `CLSTR1.QM2`. With the information from the repository, it can automatically create remote definitions for those queues. If the queues are on `QM1`, this mechanism still works, because `QM2` is a full repository. A full repository has a complete record of all the objects in the cluster. In this latter case, `QM3` would also automatically create a cluster-sender channel corresponding to the cluster-receiver channel on `QM1`, allowing direct communication between the two.

Figure 11 on page 30 shows the same cluster, with the two cluster-sender channels that were created automatically. The cluster-sender channels are represented by the two dashed lines that join with the cluster-receiver channel `CLSTR1.QM3`. It also shows the cluster transmission queue, `SYSTEM.CLUSTER.TRANSMIT.QUEUE`, which `QM1` uses to send its messages. All queue managers in the cluster have a cluster transmission queue, from which they can send messages to any other queue manager in the same cluster.

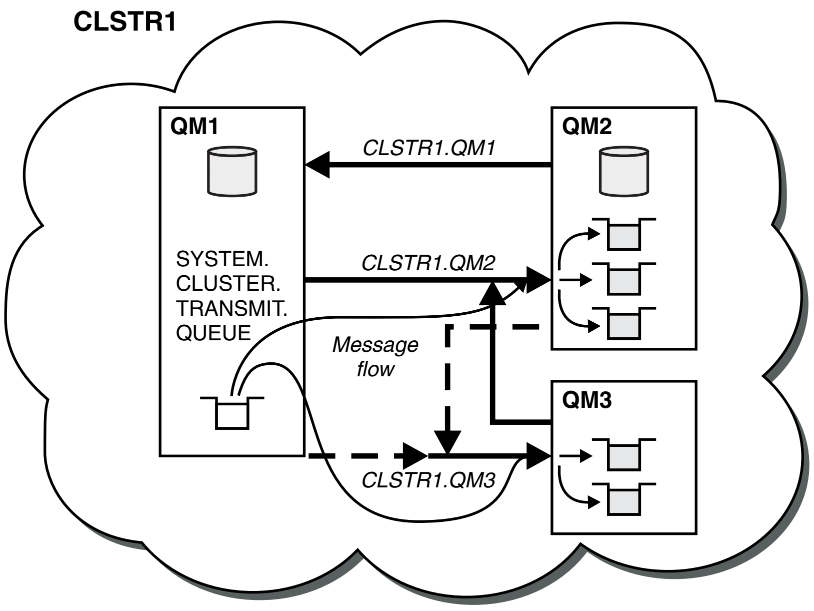


Figure 11. A cluster of queue managers, showing auto-defined channels

Note: Other diagrams show only the receiving ends of channels for which you make manual definitions. The sending ends are omitted because they are mostly defined automatically when needed. The auto-definition of most cluster-sender channels is crucial to the function and efficiency of clusters.

Related concepts:

"Comparison of clustering and distributed queuing" on page 17
Compare the components that need to be defined to connect queue managers using distributed queuing and clustering.

Related information:

Configuring a queue manager cluster

Components of a cluster

Setting up a new cluster

Clustering: Best practices:

Clusters provide a mechanism for interconnecting queue managers. The best practices described in this section are based on testing and feedback from customers.

A successful cluster setup is dependent on good planning and a thorough understanding of IBM MQ fundamentals, such as good application management and network design. Ensure that you are familiar with the information in the related topics before continuing.

Related tasks:

"Designing clusters" on page 11
Clusters provide a mechanism for interconnecting queue managers in a way that simplifies both the initial configuration and the ongoing management. Clusters must be carefully designed, to ensure that they function correctly, and that they achieve the required levels of availability and responsiveness.

Related information:

Distributed queuing and clusters

Clusters

Monitoring clusters

Clustering: Special considerations for overlapping clusters:

This topic provides guidance for planning and administering IBM MQ clusters. This information is a guide based on testing and feedback from customers.
- "Cluster ownership"
- "Overlapping clusters: Gateways"
- "Cluster naming conventions" on page 32

Cluster ownership

Familiarize yourself with overlapping clusters before reading the following information. See "Overlapping clusters" on page 23 and Configuring message paths between clusters for the necessary information.

When configuring and managing a system that consists of overlapping clusters, it is best to adhere to the following:
- Although IBM MQ clusters are 'loosely coupled' as previously described, it is useful to consider a cluster as a single unit of administration. This concept is used because the interaction between definitions on individual queue managers is critical to the smooth functioning of the cluster. For example: When using workload balanced cluster queues it is important that a single administrator or team understand the full set of possible destinations for messages, which depends on definitions spread throughout the cluster. More trivially, cluster sender/receiver channel pairs must be compatible throughout.
- Considering this previous concept; where multiple clusters meet (which are to be administered by separate teams / individuals), it is important to have clear policies in place controlling administration of the gateway queue managers.
- It is useful to treat overlapping clusters as a single namespace: Channel names and queue manager names must be unique throughout a single cluster. Administration is much easier when unique throughout the entire topology. It is best to follow a suitable naming convention, possible conventions are described in "Cluster naming conventions" on page 32.
- Sometimes administrative and system management cooperation is essential/unavoidable: For example, cooperation between organizations that own different clusters that need to overlap. A clear understanding of who owns what, and enforceable rules/conventions helps clustering run smoothly when overlapping clusters.

Overlapping clusters: Gateways

In general, a single cluster is easier to administer than multiple clusters. Therefore creating large numbers of small clusters (one for every application for example) is something to be avoided generally.

However, to provide classes of service, you can implement overlapping clusters. For example:
- Concentric clusters where the smaller one is for Publish/Subscribe. See How to size systems: for more information.

- Some queue managers are to be administered by different teams (see "Cluster ownership" on page 31).
- If it makes sense from an organizational or geographical point of view.
- Equivalent clusters to work with name resolution, as when implementing TLS in an existing cluster.

There is no security benefit from overlapping clusters; allowing clusters administered by two different teams to overlap, effectively joins the teams as well as the topology. Any:
- Name advertised in such a cluster is accessible to the other cluster.
- Name advertised in one cluster can be advertised in the other to draw off eligible messages.
- Non-advertised object on a queue manager adjacent to the gateway can be resolved from any clusters of which the gateway is a member.

The namespace is the union of both clusters and must be treated as a single namespace. Therefore, ownership of an overlapping cluster is shared amongst all the administrators of both clusters.

When a system contains multiple clusters, there might be a requirement to route messages from queue managers in one cluster to queues on queue managers in another cluster. In this situation, the multiple clusters must be interconnected in some way: A good pattern to follow is the use of gateway queue managers between clusters. This arrangement avoids building up a difficult-to-manage mesh of point-to-point channels, and provides a good place to manage such issues as security policies. There are two distinct ways of achieving this arrangement:
1. Place one (or more) queue managers in both clusters using a second cluster receiver definition. This arrangement involves fewer administrative definitions but, as previously stated, means that ownership of an overlapping cluster is shared amongst all the administrators of both clusters.
2. Pair a queue manager in cluster one with a queue manager in cluster teo using traditional point-to-point channels.

In either of these cases, various tools can be used to route traffic appropriately. In particular, queue or queue manager aliases can be used to route into the other cluster, and a queue manager alias with blank **RQMNAME** property re-drives workload balancing where it is wanted.

Cluster naming conventions

This information contains the previous guidance on naming conventions, and the current guidance. As the IBM MQ technology improves, and as customers use technology in new or different ways, new recommendations and information must be provided for these scenarios.

Cluster naming conventions: Previous guidance

When setting up a new cluster, consider a naming convention for the queue managers. Every queue manager must have a different name, but it might help you to remember which queue managers are grouped where if you give them a set of similar names.

Every cluster-receiver channel must also have a unique name.

If you have more than one channel to the same queue manager, each with different priorities or using different protocols, you might extend the names to include the different protocols; for example `QM1.S1`, `QM1.N3`, and `QM1.T4`. In this example, `S1` might be the first SNA channel, `N3` might be the NetBIOS channel with a network priority of 3.

The final qualifier might describe the class of service the channel provides. For more information, see Defining classes of service.

Remember that all cluster-sender channels have the same name as their corresponding cluster-receiver channel.

Do not use generic connection names on your cluster-receiver definitions. In IBM MQ for z/OS, you can define VTAM generic resources or *Dynamic Domain Name Server* (DDNS) generic names, but do not do this if you are using clusters. If you define a CLUSRCVR with a generic **CONNAME**, there is no guarantee that your CLUSSDR channels point to the queue managers that you intend. Your initial CLUSSDR might end up pointing to any queue manager in the queue-sharing group, not necessarily one that hosts a full repository. Furthermore, if a channel goes to retry status, it might reconnect to a different queue manager with the same generic name and the flow of your messages is disrupted.

Cluster naming conventions: Current guidance

The previous guidance in the section, "Cluster naming conventions: Previous guidance" on page 32, is still valid. However the following guidance is intended as an update when designing new clusters. This updated suggestion ensures uniqueness of channels across multiple clusters, allowing multiple clusters to be successfully overlapped. Because queue managers and clusters can have names of up to 48 characters, and a channel name is limited to 20 characters, care must be taken when naming objects from the beginning to avoid having to change the naming convention midway through a project.

When setting up a new cluster, consider a naming convention for the queue managers. Every queue manager must have a different name. If you give queue managers in a cluster a set of similar names, it might help you to remember which queue managers are grouped where.

When defining channels, remember that all automatically created cluster-sender channels on any queue manager in the cluster have the same name as their corresponding cluster-receiver channel configured on the receiving queue manager in the cluster, and must therefore be unique and make sense across the cluster to the administrators of that cluster. Channel names are limited to a maximum of 20 characters.

One possibility is to use the queue manager name preceded by the cluster-name. For example, if the cluster-name is CLUSTER1 and the queue managers are QM1, QM2, then cluster-receiver channels are CLUSTER1.QM1, CLUSTER1.QM2.

You might extend this convention if channels have different priorities or use different protocols; for example, `CLUSTER1.QM1.S1`, `CLUSTER1.QM1.N3`, and `CLUSTER1.QM1.T4`. In this example, `S1` might be the first SNA channel, `N3` might be the NetBIOS channel with a network priority of three.

A final qualifier might describe the class of service the channel provides.

z/OS In IBM MQ for z/OS, you can define VTAM generic resources or *Dynamic Domain Name Server* (DDNS) generic names. You can define connection names using generic names. However, when you create a cluster-receiver definition, do not use a generic connection name.

z/OS The problem with using generic connection names for cluster-receiver definitions is as follows. If you define a `CLUSRCVR` with a generic `CONNAME` there is no guarantee that your `CLUSSDR` channels point to the queue managers you intend. Your initial `CLUSSDR` might end up pointing to any queue manager in the queue-sharing group, not necessarily one that hosts a full repository. If a channel starts trying a connection again, it might reconnect to a different queue manager with the same generic name disrupting the flow of messages.

Clustering: Topology design considerations:

This topic provides guidance for planning and administering IBM MQ clusters. This information is a guide based on testing and feedback from customers.

By thinking about where user applications and internal administrative processes are going to be located in advance, many problems can either be avoided, or minimized at a later date. This topic contains information about design decisions that can improve performance, and simplify maintenance tasks as the cluster scales.

- "Performance of the clustering infrastructure"
- "Full repositories" on page 35
- "Should applications use queues on full repositories?" on page 36
- "Managing channel definitions" on page 36
- "Workload balancing over multiple channels" on page 36

Performance of the clustering infrastructure

When an application tries to open a queue on a queue manager in a cluster, the queue manager registers its interest with the full repositories for that queue so that it can learn where the queue exists in the cluster. Any updates to the queue location or configuration are automatically sent by the full repositories to the interested queue manager. This registering of interest is internally known as a subscription (these subscriptions are not the same as IBM MQ subscriptions used for publish/subscribe messaging in IBM MQ)

All information about a cluster goes through every full repository. Full repositories are therefore always being used in a cluster for administrative message traffic. The high usage of system resources when managing these subscriptions, and the transmission of them and the resulting configuration messages, can cause a considerable load on the clustering infrastructure. There are a number of things to consider when ensuring that this load is understood and minimized wherever possible:

- The more individual queue managers using a cluster queue, the more subscriptions are in the system, and thus the bigger the administrative overhead when changes occur and interested subscribers need to be notified, especially on the full repository queue managers. One way to minimize unnecessary traffic and full repository load is by connecting similar applications (that is, those applications that work with the same queues) to a smaller number of queue managers.
- In addition to the number of subscriptions in the system affecting the performance the rate of change in the configuration of clustered objects can affect performance, for example the frequent changing of a clustered queue configuration.
- When a queue manager is a member of multiple clusters (that is, it is part of an overlapping cluster system) any interest made in a queue results in a subscription for each cluster it is a member of, even if the same queue managers are the full repositories for more than one of the clusters. This arrangement increases the load on the system, and is one reason to consider whether multiple overlapping clusters are necessary, rather than a single cluster.
- Application message traffic (that is, the messages being sent by IBM MQ applications to the cluster queues) does not go via the full repositories to reach the destination queue managers. This message traffic is sent directly between the queue manager where the message enters the cluster, and the queue manager where the cluster queue exists. It is not therefore necessary to accommodate high rates of application message traffic with respect to the full repository queue managers, unless the full repository queue managers happen to be either of those two queue managers mentioned. For that reason, it is recommended that full repository queue managers are not used for application message traffic in clusters where the clustering infrastructure load is significant.

Full repositories

A repository is a collection of information about the queue managers that are members of a cluster. A queue manager that hosts a complete set of information about every queue manager in the cluster has a full repository. For more information about full repositories and partial repositories, see Cluster repository.

Full repositories must be held on servers that are reliable and as highly available as possible and single points of failure must be avoided. The cluster design must always have two full repositories. If there is a failure of a full repository, the cluster can still operate.

Details of any updates to cluster resources made by a queue manager in a cluster; for example, clustered queues, are sent from that queue manager to two full repositories at most in that cluster (or to one if there is only one full repository queue manager in the cluster). Those full repositories hold the information and propagate it to any queue managers in the cluster that show an interest in it (that is, they subscribe to it). To ensure that each member of the cluster has an up-to-date view of the cluster resources there, each queue manager must be able to communicate with at least one full repository queue manager at any one time.

If, for any reason a queue manager cannot communicate with any full repositories, it can continue to function in the cluster based on its already cached level of information for a period time, but no new updates or access to previously unused cluster resources are available.

For this reason, you must aim to keep the two full repositories available at all times. However, this arrangement does not mean that extreme measures must be taken because the cluster functions adequately for a short while without a full repository.

There is another reason that a cluster must have two full repository queue managers, other than the availability of cluster information: This reason is to ensure that the cluster information held in the full repository cache exists in two places for recovery purposes. If there is only one full repository, and it loses its information about the cluster, then manual intervention on all queue managers within the cluster is required in order to get the cluster working again. If there are two full repositories however, then because information is always published to and subscribed for from two full repositories, the failed full repository can be recovered with the minimum of effort.

- It is possible to perform maintenance on full repository queue managers in a two full repository cluster design without impacting users of that cluster: The cluster continues to function with only one repository, so where possible bring the repositories down, apply the maintenance, and back up again one at a time. Even if there is an outage on the second full repository, running applications are unaffected for a minimum of three days.
- Unless there is a good reason for using a third repository, such as using a geographically local full repository for geographical reasons, use the two repository design. Having three full repositories means that you never know which are the two that are currently in use, and there might be administrative problems caused by interactions between multiple workload management parameters. It is not recommend to have more than two full repositories.
- If you still need better availability, consider hosting the full repository queue managers as multi-instance queue managers or using platform specific high availability support to improve their availability.
- You must fully interconnect all the full repository queue managers with manually defined cluster sender channels. Particular care must be taken when the cluster does have, for some justifiable reason, more than two full repositories. In this situation it is often possible to miss one or more channels and for it not to be immediately apparent. When full interconnection does not occur, hard to diagnose problems often arise. They are hard to diagnose because some full repositories not holding all repository data and therefore resulting in queue managers in the cluster having different views of the cluster depending on the full repositories that they connect to.

Should applications use queues on full repositories?

A full repository is in most ways exactly like any other queue manager, and it is therefore possible to host application queues on the full repository and connect applications directly to these queue managers. Should applications use queues on full repositories?

The commonly accepted answer is "No?". Although this configuration is possible, many customers prefer to keep these queue managers dedicated to maintaining the full repository cluster cache. Points to consider when deciding on either option are described here, but ultimately the cluster architecture must be appropriate to the particular demands of the environment.

- Upgrades: Usually, in order to use new cluster features in new releases of IBM MQ the full repository queue managers of that cluster must be upgraded first. When an application in the cluster wants to use new features, it might be useful to be able to update the full repositories (and some subset of partial repositories) without testing a number of co-located applications.
- Maintenance: In a similar way if you must apply urgent maintenance to the full repositories, they can be restarted or refreshed with the `REFRESH` command without touching applications.
- Performance: As clusters grow and demands on the full repository cluster cache maintenance become greater, keeping applications separate reduces risk of this affecting application performance through contention for system resources.
- Hardware requirements: Typically, full repositories do not need to be powerful; for example, a simple UNIX server with a good expectation of availability is sufficient. Alternatively, for very large or constantly changing clusters, the performance of the full repository computer must be considered.
- Software requirements: Requirements are usually the main reason for choosing to host application queues on a full repository. In a small cluster, collocation might mean a requirement for fewer queue managers/servers over all.

Managing channel definitions

Even within a single cluster, multiple channel definitions can exist giving multiple routes between two queue managers.

There is sometimes an advantage to having parallel channels within a single cluster, but this design decision must be considered thoroughly; apart from adding complexity, this design might result in channels being under-used which reduces performance. This situation occurs because testing usually involves sending lots of messages at a constant rate, so the parallel channels are fully used. But with real-world conditions of a non-constant stream of messages, the workload balancing algorithm causes performance to drop as the message flow is switched from channel to channel.

When a queue manager is a member of multiple clusters, the option exists to use a single channel definition with a cluster namelist, rather than defining a separate CLUSRCVR channel for each cluster. However, this setup can cause administration difficulties later; consider for example the case where TLS is to be applied to one cluster but not a second. It is therefore preferable to create separate definitions, and the naming convention suggested in “Cluster naming conventions” on page 32 supports this.

Workload balancing over multiple channels

This information is intended as an advanced understanding of the subject. For the basic explanation of this subject (which must be understood before using the information here), see Using clusters for workload management, Workload balancing in clusters, and The cluster workload management algorithm.

The cluster workload management algorithm provides a large set of tools, but they must not all be used with each other without fully understanding how they work and interact. It might not be immediately obvious how important channels are to the workload balancing process: The workload management round-robin algorithm behaves as though multiple cluster channels to a queue manager that owns a clustered queue, are treated as multiple instances of that queue. This process is explained in more detail in the following example:

1. There are two queue managers hosting a queue in a cluster: `QM1` and `QM2`.
2. There are five cluster receiver channels to `QM1`.
3. There is only one cluster receiver channel to `QM2`.
4. When **`MQPUT`** or **`MQOPEN`** on `QM3` chooses an instance, the algorithm is five times more likely to send the message to `QM1` than to `QM2`.
5. The situation in step 4 occurs because the algorithm sees six options to choose from (5+1) and round-robins across all five channels to `QM1` and the single channel to `QM2`.

Another subtle behavior is that even when putting messages to a clustered queue that happens to have one instance configured on the local queue manager, IBM MQ uses the state of the local cluster receiver channel to decide whether messages are to be put to the local instance of the queue or remote instances of the queue. In this scenario:

1. When putting messages the workload management algorithm does not look at individual cluster queues, it looks at the cluster channels which can reach those destinations.
2. To reach local destinations, the local receiver channels are included in this list (although they are not used to send the message).
3. When a local receiver channel is stopped, the workload management algorithm, prefers an alternative instance by default if its CLUSRCVR is not stopped. If there are multiple local CLUSRCVR instances for the destination and at least one is not stopped, the local instance remains eligible.

Clustering: Application isolation using multiple cluster transmission queues:

You can isolate the message flows between queue managers in a cluster. You can place messages being transported by different cluster-sender channels onto different cluster transmission queues. You can use the approach in a single cluster or with overlapping clusters. The topic provides examples and some best practices to guide you in choosing an approach to use.

When you deploy an application, you have a choice over which IBM MQ resources it shares with other applications and which resources it does not share. There are a number of types of resources that can be shared, the main ones being the server itself, the queue manager, channels, and queues. You can choose to configure applications with fewer shared resources; allocating separate queues, channels, queue managers, or even servers to individual applications. If you do so, the overall system configuration becomes bigger and more complex. Using IBM MQ clusters reduces the complexity of managing more servers, queue managers, queues, and channels, but it introduces another shared resource, the cluster transmission queue, `SYSTEM.CLUSTER.TRANSMIT.QUEUE`.

Figure 12 on page 39 is a slice out of a large IBM MQ deployment that illustrates the significance of sharing `SYSTEM.CLUSTER.TRANSMIT.QUEUE`. In the diagram, the application, `Client App`, is connected to the queue manager `QM2` in cluster `CL1`. A message from `Client App` is processed by the application, `Server App`. The message is retrieved by `Server App` from the cluster queue `Q1` on the queue manager `QM3` in `CLUSTER2`. Because the client and server applications are not in the same cluster, the message is transferred by the gateway queue manager `QM1`.

The normal way to configure a cluster gateway is to make the gateway queue manager a member of all the clusters. On the gateway queue manager are defined clustered alias queues for cluster queues in all the clusters. The clustered queue aliases are available in all the clusters. Messages put to the cluster queue aliases are routed via the gateway queue manager to their correct destination. The gateway queue manager puts messages sent to the clustered alias queues onto the common `SYSTEM.CLUSTER.TRANSMIT.QUEUE` on `QM1`.

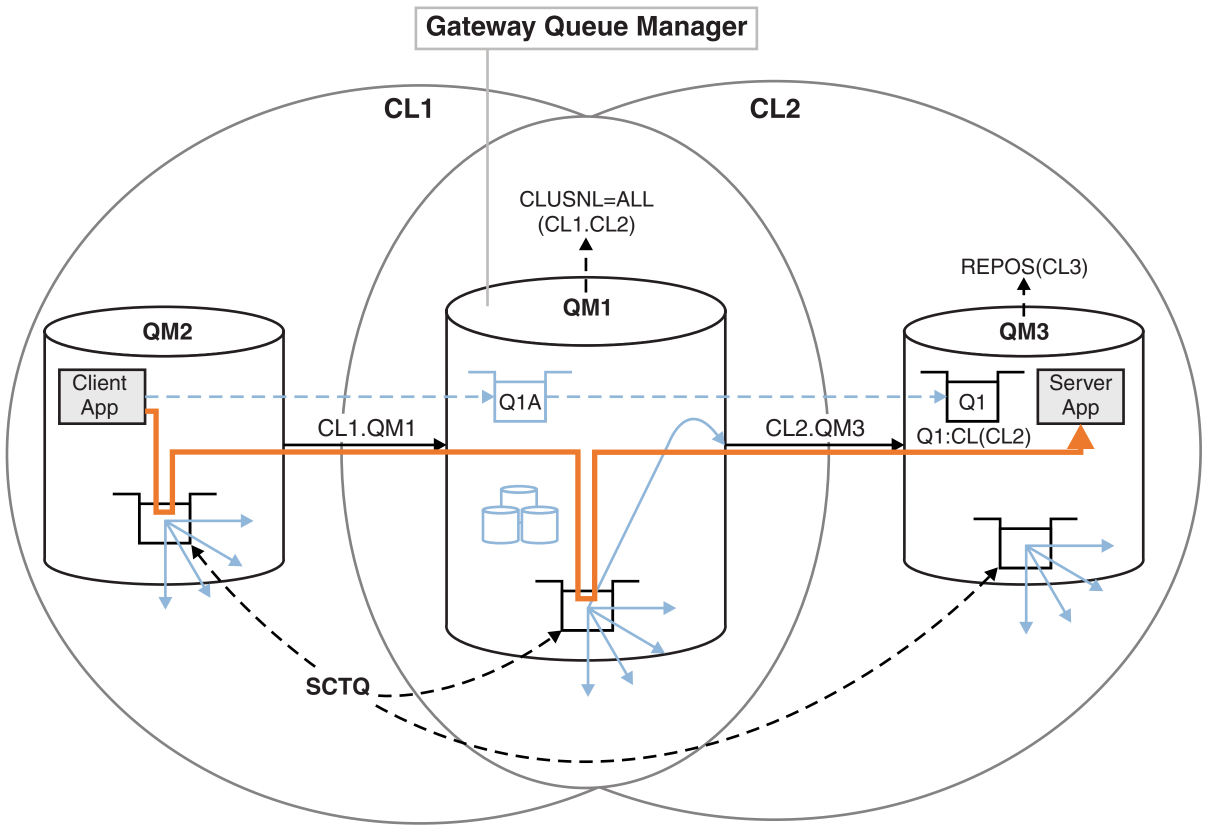
The hub and spoke architecture requires all messages between clusters to pass through the gateway queue manager. The result is that all messages flow through the single cluster transmission queue on `QM1`, `SYSTEM.CLUSTER.TRANSMIT.QUEUE`.

From a performance perspective, a single queue is not a problem. A common transmission queue generally does not represent a performance bottleneck. Message throughput on the gateway is largely determined by the performance of the channels that connect to it. Throughput is not generally affected by the number of queues, or the number of messages on the queues that use the channels.

From some other perspectives, using a single transmission queue for multiple applications has drawbacks:

- You cannot isolate the flow of messages to one destination from the flow of messages to another destination. You cannot separate the storage of messages before they are forwarded, even if the destinations are in different clusters on different queue managers.

  If one cluster destination becomes unavailable, messages for that destination build-up in the single transmission queue, and eventually the messages fill it up. Once the transmission queue is full, it stops messages from being placed onto the transmission queue for any cluster destination.
- It is not easy to monitor the transfer of messages to different cluster destinations. All the messages are on the single transmission queue. Displaying the depth of the transmission queue gives you little indication whether messages are being transferred to all destinations.



Note: The arrows in Figure 12 and following figures are of different types. Solid arrows represent message flows. The labels on solid arrows are message channel names. The gray solid arrows are potential message flows from the SYSTEM.CLUSTER.TRANSMIT.QUEUE onto cluster-sender channels. Black dashed lines connect labels to their targets. Gray dashed arrows are references; for example from an MQOPEN call by Client App to the cluster alias queue definition Q1A.

Figure 12. Client-server application deployed to hub and spoke architecture using IBM MQ clusters

In Figure 12, clients of Server App open the queue Q1A. Messages are put to SYSTEM.CLUSTER.TRANSMIT.QUEUE on QM2, transferred to SYSTEM.CLUSTER.TRANSMIT.QUEUE on QM1, and then transferred to Q1 on QM3, where they are received by the Server App application.

The message from Client App passes through system cluster transmission queues on QM2 and QM1. In Figure 12, the objective is to isolate the message flow on the gateway queue manager from the client application, so that its messages are not stored on SYSTEM.CLUSTER.TRANSMIT.QUEUE. You can isolate flows on any of the other clustered queue managers. You can also isolate flows in the other direction, back to the client. To keep the descriptions of the solutions brief, the descriptions consider only a single flow from the client application.

Solutions to isolating cluster message traffic on a cluster gateway queue manager

One way to solve the problem is to use queue manager aliases, or remote queue definitions, to bridge between clusters. Create a clustered remote queue definition, a transmission queue, and a channel, to separate each message flow on the gateway queue manager; see Adding a remote queue definition to isolate messages sent from a gateway queue manager.

From Version 7.5 onwards, cluster queue managers are not limited to a single cluster transmission queue. You have two choices:

1. Define additional cluster transmission queues manually, and define which cluster-sender channels transfer messages from each transmission queue; see Adding a cluster transmit queue to isolate cluster message traffic sent from a gateway queue manager.
2. Allow the queue manager to create and manage additional cluster transmission queues automatically. It defines a different cluster transmission queue for each cluster-sender channel; see Changing the default to separate cluster transmission queues to isolate message traffic.

You can combine manually defined cluster transmission queues for some cluster-sender channels, with the queue manager managing the rest. The combination of transmission queues is the approach taken in Adding a cluster transmit queue to isolate cluster message traffic sent from a gateway queue manager. In that solution, most messages between clusters use the common `SYSTEM.CLUSTER.TRANSMIT.QUEUE`. One application is critical, and all its message flows are isolated from other flows by using one manually defined cluster transmission queue.

The configuration in Adding a cluster transmit queue to isolate cluster message traffic sent from a gateway queue manager is limited. It does not separate the message traffic going to a cluster queue on the same queue manager in the same cluster as another cluster queue. You can separate the message traffic to individual queues by using the remote queue definitions that are part of distributed queuing. With clusters, using multiple cluster transmission queues, you can separate message traffic that goes to different cluster-sender channels. Multiple cluster queues in the same cluster, on the same queue manager, share a cluster-sender channel. Messages for those queues are stored on the same transmission queue, before being forwarded from the gateway queue manager. In the configuration in Adding a cluster and a cluster transmit queue to isolate cluster message traffic sent from a gateway queue manager, the limitation is side-stepped by adding another cluster and making the queue manager and cluster queue a member of the new cluster. The new queue manager might be the only queue manager in the cluster. You could add more queue managers to the cluster, and use the same cluster to isolate cluster queues on those queue managers as well.

Related concepts:

"Access control and multiple cluster transmission queues" on page 16
Choose between three modes of checking when an application puts messages to remote cluster queues. The modes are checking remotely against the cluster queue, checking locally against `SYSTEM.CLUSTER.TRANSMIT.QUEUE`, or checking against local profiles for the cluster queue, or cluster queue manager.

"Overlapping clusters" on page 23
Overlapping clusters provide additional administrative capabilities. Use namelists to reduce the number of commands needed to administer overlapping clusters.

Related information:

Authorizing putting messages on remote cluster queues

Working with cluster transmission queues and cluster-sender channels

Adding a remote queue definition to isolate messages sent from a gateway queue manager

Adding a cluster transmit queue to isolate cluster message traffic sent from a gateway queue manager

Adding a cluster and a cluster transmit queue to isolate cluster message traffic sent from a gateway queue manager

Changing the default to separate cluster transmission queues to isolate message traffic

Creating two-overlapping clusters with a gateway queue manager

Configuring message paths between clusters

Securing

setmqaut

Clustering: Planning how to configure cluster transmission queues:

You are guided through the choices of cluster transmission queues. You can configure one common default queue, separate default queues, or manually defined queues. z/OS To configure a cluster-sender channel to use a transmission queue other than SYSTEM.CLUSTER.TRANSMIT.QUEUE, you need to enable new function, using the mode of operation (OPMODE) system parameter in the CSQ6SYSP parameter.

Before you begin

Review "How to choose what type of cluster transmission queue to use" on page 43.

z/OS See the mode of operation (OPMODE) topic for more information.

About this task

You have some choices to make when you are planning how to configure a queue manager to select a cluster transmission queue.

1. What is the default cluster transmission queue for cluster message transfers?
   a. A common cluster transmission queue, `SYSTEM.CLUSTER.TRANSMIT.QUEUE`.
   b. Separate cluster transmission queues. The queue manager manages the separate cluster transmission queues. It creates them as permanent-dynamic queues from the model queue, `SYSTEM.CLUSTER.TRANSMIT.MODEL.QUEUE`. It creates one cluster transmission queue for each cluster-sender channel it uses.
2. For the cluster transmission queues that you do decide to create manually, you have a further two choices:
   a. Define a separate transmission queue for each cluster-sender channel that you decide to configure manually. In this case, set the **`CLCHNAME`** queue attribute of the transmission queue to the name of a cluster-sender channel. Select the cluster-sender channel that is to transfer messages from this transmission queue.
   b. Combine message traffic for a group of cluster-sender channels onto the same cluster transmission queue; see Figure 13 on page 42. In this case, set the **`CLCHNAME`** queue attribute of each common transmission queue to a generic cluster-sender channel name. A generic cluster-sender channel name is a filter to group cluster-sender channel names. For example, `SALES.*` groups all cluster-sender channels that have names beginning with `SALES.`. You can place multiple wildcard characters anywhere in the filter-string. The wildcard character is an asterisk, "*". It represents from zero to any number of characters.

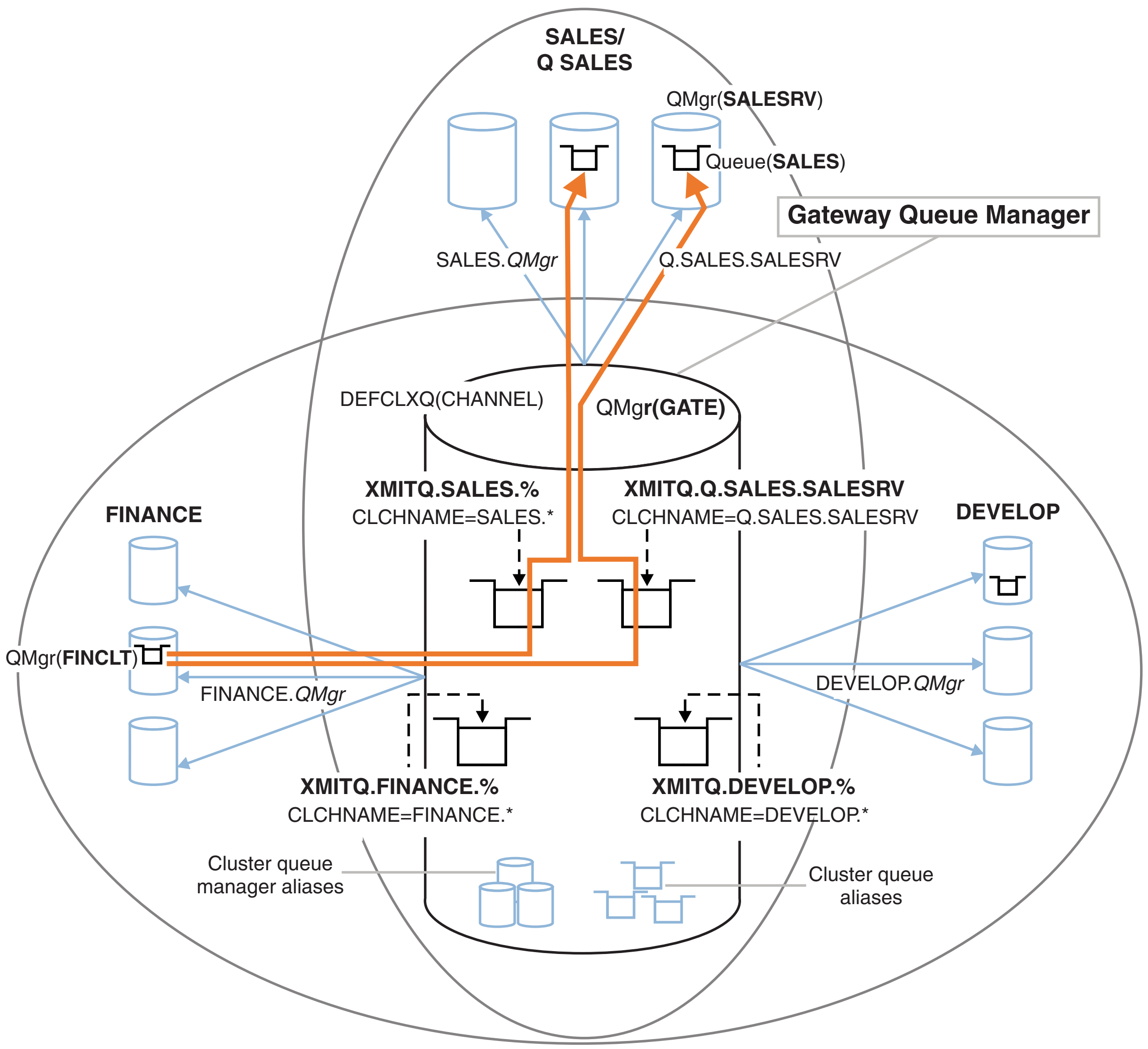


Figure 13. Example of specific transmission queues for different departmental IBM MQ clusters

Procedure

1. Select the type of default cluster transmission queue to use.
   - Choose a single cluster transmission queue, or separate queues for each cluster connection.

   Leave the default setting or run the **MQSC** command:

   ```
   ALTER QMGR DEFCLXQ(CHANNEL)
   ```
2. Isolate any message flows that must not share a cluster transmission queue with other flows.
   - See "Clustering: Example configuration of multiple cluster transmission queues" on page 45. In the example the `SALES` queue, which must be isolated, is a member of the `SALES` cluster, on `SALESRV`. To isolate the `SALES` queue, create a new cluster `Q.SALES`, make the `SALESRV` queue manager a member, and modify the `SALES` queue to belong to `Q.SALES`.
   - Queue managers that send messages to `SALES` must also be members of the new cluster. If you use a clustered queue alias and a gateway queue manager, as in the example, in many cases you can limit the changes to making the gateway queue manager a member of the new cluster.
   - However, separating flows from the gateway to the destination does not separate flows to the gateway from the source queue manager. But it sometimes turns out to be sufficient to separate flows from the gateway and not flows to the gateway. If it is not sufficient, then add the source

queue manager into the new cluster. If you want messages to travel through the gateway, move the cluster alias to the new cluster and continue to send messages to the cluster alias on the gateway, and not directly to the target queue manager.

Follow these steps to isolate message flows:

a. Configure the destinations of the flows so that each target queue is the only queue in a specific cluster, on that queue manager.
b. Create the cluster-sender and cluster-receiver channels for any new clusters you have created following a systematic naming convention.
   - See "Clustering: Special considerations for overlapping clusters" on page 31.
c. Define a cluster transmission queue for each isolated destination on every queue manager that sends messages to the target queue.
   - A naming convention for cluster transmission queues is to use the value of the cluster channel name attribute, `CLCHNAME`, prefixed with `XMITQ.`

3. Create cluster transmission queues to meet governance or monitoring requirements.
   - Typical governance and monitoring requirements result in a transmission queue per cluster or a transmission queue per queue manager. If you follow the naming convention for cluster channels, *`ClusterName. QueueManagerName`*, it is easy to create generic channel names that select a cluster of queue managers, or all the clusters a queue manager is a member of; see "Clustering: Example configuration of multiple cluster transmission queues" on page 45.
   - Extend the naming convention for cluster transmission queues to cater for generic channel names, by replacing the asterisk symbol by a percent sign. For example,

```
DEFINE QLOCAL(XMITQ.SALES.%) USAGE(XMITQ) CLCHNAME(SALES.*)
```

Related concepts:

"Access control and multiple cluster transmission queues" on page 16
Choose between three modes of checking when an application puts messages to remote cluster queues. The modes are checking remotely against the cluster queue, checking locally against `SYSTEM.CLUSTER.TRANSMIT.QUEUE`, or checking against local profiles for the cluster queue, or cluster queue manager.

"Overlapping clusters" on page 23
Overlapping clusters provide additional administrative capabilities. Use namelists to reduce the number of commands needed to administer overlapping clusters.

Related information:

Working with cluster transmission queues and cluster-sender channels

Adding a remote queue definition to isolate messages sent from a gateway queue manager

Adding a cluster transmit queue to isolate cluster message traffic sent from a gateway queue manager

Adding a cluster and a cluster transmit queue to isolate cluster message traffic sent from a gateway queue manager

Changing the default to separate cluster transmission queues to isolate message traffic

Creating two-overlapping clusters with a gateway queue manager

Configuring message paths between clusters

How to choose what type of cluster transmission queue to use:

How to choose between different cluster transmission queue configuration options.

From Version 7.5 onwards, you can choose which cluster transmission queue is associated with a cluster-sender channel.

1. You can have all cluster-sender channels associated with the single default cluster transmit queue, `SYSTEM.CLUSTER.TRANSMIT.QUEUE`. This option is the default, and is the only choice for queue managers that run version Version 7.1, or earlier.

2. You can set all cluster-sender channels to be automatically associated with a separate cluster transmission queue. The queues are created by the queue manager from the model queue SYSTEM.CLUSTER.TRANSMIT.MODEL.QUEUE and named SYSTEM.CLUSTER.TRANSMIT. *ChannelName*. Channels will use their uniquely named cluster transmit queue if the queue manager attribute **DEFCLXQ** is set to CHANNEL.
3. You can set specific cluster-sender channels to be served by a single cluster transmission queue. Select this option by creating a transmission queue and setting its **CLCHNAME** attribute to the name of the cluster-sender channel.
4. You can select groups of cluster-sender channels to be served by a single cluster transmission queue. Select this option by creating a transmission queue and setting its **CLCHNAME** attribute to a generic channel name, such as *ClusterName*.*. If you name cluster channels by following the naming conventions in "Clustering: Special considerations for overlapping clusters" on page 31, this name selects all cluster channels connected to queue managers in the cluster *ClusterName*.

You can combine either of the default cluster transmission queue options for some cluster-sender channels, with any number of specific and generic cluster transmission queue configurations.

Best practices

In most cases, for existing IBM MQ installations, the default configuration is the best choice. A cluster queue manager stores cluster messages on a single cluster transmission queue, SYSTEM.CLUSTER.TRANSMIT.QUEUE. You have the choice of changing the default to storing messages for different queue managers and different clusters on separate transmission queues, or of defining your own transmission queues.

In most cases, for new IBM MQ installations, the default configuration is also the best choice. The process of switching from the default configuration to the alternative default of having one transmission queue for each cluster-sender channel is automatic. Switching back is also automatic. The choice of one or the other is not critical, you can reverse it.

The reason for choosing a different configuration is more to do with governance, and management, than with functionality or performance. With a couple of exceptions, configuring multiple cluster transmission queues does not benefit the behavior of the queue manager. It results in more queues, and requires you to modify monitoring and management procedures you have already set up that refer to the single transmission queue. That is why, on balance, remaining with the default configuration is the best choice, unless you have strong governance or management reasons for a different choice.

The exceptions are both concerned with what happens if the number of messages stored on SYSTEM.CLUSTER.TRANSMIT.QUEUE increases. If you take every step to separate the messages for one destination from the messages for another destination, then channel and delivery problems with one destination ought not to affect the delivery to another destination. However, the number of messages stored on SYSTEM.CLUSTER.TRANSMIT.QUEUE can increase due to not delivering messages fast enough to one destination. The number of messages on SYSTEM.CLUSTER.TRANSMIT.QUEUE for one destination can affect the delivery of messages to other destinations.

To avoid problems that result from filling up a single transmission queue, aim to build sufficient capacity into your configuration. Then, if a destination fails and a message backlog starts to build up, you have time to fix the problem.

If messages are routed through a hub queue manager, such as a cluster gateway, they share a common transmission queue, SYSTEM.CLUSTER.TRANSMIT.QUEUE. If the number of messages stored on SYSTEM.CLUSTER.TRANSMIT.QUEUE on the gateway queue manager reaches its maximum depth, the queue manager starts to reject new messages for the transmission queue until its depth reduces. The congestion affects messages for all destinations that are routed through the gateway. Messages back up the transmission queues of other queue managers that are sending messages to the gateway. The problem manifests itself in messages written to queue manager error logs, falling message throughput, and longer elapsed times between sending a message and the time that a message arrives at its destination.

The effect of congestion on a single transmission queue can become apparent, even before it is full. If you have a mixed message traffic, with some large non-persistent messages and some small messages, the time to deliver small messages increases as the transmission queue fills. The delay is due to writing large non-persistent messages to disk that would not normally get written to disk. If you have time critical message flows, sharing a cluster transmission queue with other mixed messages flows, it could be worth configuring a special message path to isolate it from other message flows; see Adding a cluster and a cluster transmit queue to isolate cluster message traffic sent from a gateway queue manager.

The other reasons for configuring separate cluster transmission queues are to meet governance requirements, or to simplify monitoring messages that are sent to different cluster destinations. For example, you might have to demonstrate that messages for one destination never share a transmission queue with messages for another destination.

Change the queue manager attribute **`DEFCLXQ`** that controls the default cluster transmission queue, to create different cluster transmission queues for every cluster-sender channel. Multiple destinations can share a cluster-sender channel, so you must plan your clusters to meet this objective fully. Apply the method Adding a cluster and a cluster transmit queue to isolate cluster message traffic sent from a gateway queue manager systematically to all your cluster queues. The result you are aiming for is for no cluster destination to share a cluster-sender channel with another cluster destination. As a consequence, no message for a cluster destination shares its cluster transmission queue with a message for another destination.

Creating a separate cluster transmission queue for some specific message flow, makes it easy to monitor the flow of messages to that destination. To use a new cluster transmission queue, define the queue, associate it with a cluster-sender channel, and stop and start the channel. The change does not have to be permanent. You could isolate a message flow for a while, to monitor the transmission queue, and then revert to using the default transmission queue again.

Clustering: Example configuration of multiple cluster transmission queues:

In this task you apply the steps to plan multiple cluster transmission queues to three overlapping clusters. The requirements are to separate messages flows to one cluster queue, from all other message flows, and to store messages for different clusters on different cluster transmission queues.

About this task

The steps in this task show how to apply the procedure in “Clustering: Planning how to configure cluster transmission queues” on page 41 and arrive at the configuration shown in Figure 14 on page 46. It is an example of three overlapping clusters, with a gateway queue manager, that is configured with separate cluster transmission queues. The MQSC commands to define the clusters are described in “Creating the example clusters” on page 48.

For the example, there are two requirements. One is to separate the message flow from the gateway queue manager to the sales application that logs sales. The second is to query how many messages are waiting to be sent to different departmental areas at any point in time. The `SALES`, `FINANCE`, and `DEVELOP` clusters are already defined. Cluster messages are currently forwarded from `SYSTEM.CLUSTER.TRANSMIT.QUEUE`.

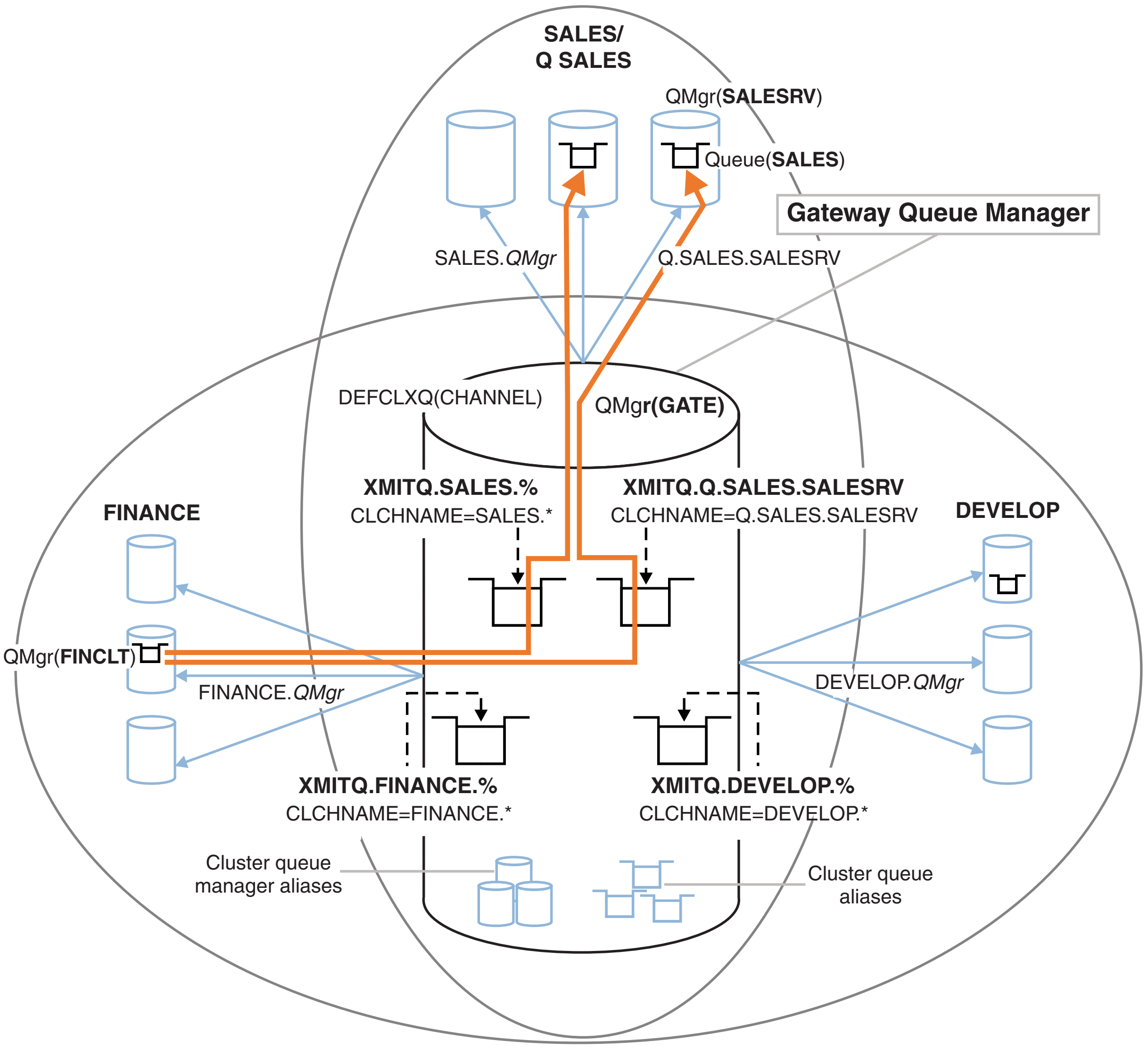


Figure 14. Example of specific transmission queues for different departmental IBM MQ clusters

The steps to modify the clusters are as follows; see Figure 16 on page 50 for the definitions.

Procedure

1. The first configuration step is to " Select the type of default cluster transmission queue to use ".

   The decision is to create separate default cluster transmission queues by running the following **MQSC** command on the GATE queue manager.

   ```
   ALTER QMGR DEFCLXQ(CHANNEL)
   ```

   There is no strong reason for choosing this default, as the intention is to manually define cluster transmission queues. The choice does have a weak diagnostic value. If a manual definition is done wrongly, and a message flows down a default cluster transmission queue, it shows up in the creation of a permanent-dynamic cluster transmission queue.
2. The second configuration step is to " Isolate any message flows that must not share a cluster transmission queue with other flows ". In this case the sales application that receives messages from the queue SALES on SALESRV requires isolation. Only isolation of messages from the gateway queue manager is required. The three sub-steps are:

   a. “ Configure the destinations of the flows so that each target queue is the only queue in a specific cluster, on that queue manager ”.

      The example requires adding the queue manager SALESRV to a new cluster within the sales department. If you have few queues that require isolation, you might decide on creating a specific cluster for the SALES queue. A possible naming convention for the cluster name is to name such clusters, Q. *QueueName*, for example Q.SALES. An alternative approach, which might be more practical if you have a large number of queues to be isolated, is to create clusters of isolated queues where and when needed. The clusters names might be QUEUES. *n*.

      In the example, the new cluster is called Q.SALES. To add the new cluster, see the definitions in Figure 16 on page 50. The summary of definition changes is as follows:

      1) Add Q.SALES to the namelist of clusters on the repository queue managers. The namelist is referred to in the queue manager **REPOSNL** parameter.
      2) Add Q.SALES to the namelist of clusters on the gateway queue manager. The namelist is referred to in all the cluster queue alias and cluster queue manager alias definitions on the gateway queue manager.
      3) Create a namelist on the queue manager SALESRV, for both the clusters it is a member of, and change the cluster membership of the SALES queue:

```
DEFINE NAMELIST(CLUSTERS) NAMES(SALES, Q.SALES) REPLACE
ALTER QLOCAL(SALES) CLUSTER(' ') CLUSNL(SALESRV.CLUSTERS)
```

         The SALES queue is a member of both clusters, just for the transition. Once the new configuration is running, you remove the SALES queue from the SALES cluster; see Figure 17 on page 50.

   b. “ Create the cluster-sender and cluster-receiver channels for any new clusters you have created following a systematic naming convention ”.

      1) Add the cluster-receiver channel Q.SALES. *RepositoryQMgr* to each of the repository queue managers
      2) Add the cluster-sender channel Q.SALES. *OtherRepositoryQMgr* to each of the repository queue managers, to connect to the other repository manager. Start these channels.
      3) Add the cluster receiver channels Q.SALES.SALESRV, and Q.SALES.GATE to either of the repository queue managers that is running.
      4) Add the cluster-sender channels Q.SALES.SALESRV, and Q.SALES.GATE to the SALESRV and GATE queue managers. Connect the cluster-sender channel to the repository queue manager that you created the cluster-receiver channels on.

   c. “ Define a cluster transmission queue for each isolated destination on every queue manager that sends messages to the target queue ”.

      On the gateway queue manager define the cluster transmission queue XMITQ.Q.SALES.SALESRV for the Q.SALES.SALESRV cluster-sender channel:

```
DEFINE QLOCAL(XMITQ.Q.SALES.SALESRV) USAGE(XMITQ) CLCHNAME(Q.SALES.SALESRV) REPLACE
```

3. The third configuration step is to “ Create cluster transmission queues to meet governance or monitoring requirements ”.

   On the gateway queue manager define the cluster transmission queues:

```
DEFINE QLOCAL(XMITQ.SALES)  USAGE(XMITQ) CLCHNAME(SALES.*)  REPLACE
DEFINE QLOCAL(XMITQ.DEVELOP) USAGE(XMITQ) CLCHNAME(DEVELOP.*) REPLACE
DEFINE QLOCAL(XMITQ.FINANCE) USAGE(XMITQ) CLCHNAME(SALES.*)  REPLACE
```

What to do next

Switch to the new configuration on the gateway queue manager.

The switch is triggered by starting the new channels, and restarting the channels that are now associated with different transmission queues. Alternatively, you can stop and start the gateway queue manager.

1. Stop the following channels on the gateway queue manager:
   ```
   SALES. Qmgr
   DEVELOP. Qmgr
   FINANCE. Qmgr
   ```
2. Start the following channels on the gateway queue manager:
   ```
   SALES. Qmgr
   DEVELOP. Qmgr
   FINANCE. Qmgr
   Q.SALES.SAVESRV
   ```

When the switch is complete, remove the `SALES` queue from the `SALES` cluster; see Figure 17 on page 50.

Creating the example clusters:

The definitions and instructions to create the example cluster, and modify it to isolate the `SALES` queue and separate messages on the gateway queue manager.

About this task

The full **`MQSC`** commands to create the `FINANCE`, `SALES`, and `Q.SALES` clusters are provided in Figure 15 on page 49, Figure 16 on page 50, and Figure 17 on page 50. The definitions for the basic clusters are in Figure 15 on page 49. The definitions in Figure 16 on page 50 modify the basic clusters to isolate the `SALES` queue, and to separate cluster messages to `FINANCE` and `SALES`. Finally, to remove the `SALES` queue from the `SALES` cluster; see Figure 17 on page 50. The `DEVELOP` cluster is omitted from the definitions, to keep the definitions shorter.

Procedure

1. Create the `SALES` and `FINANCE` clusters, and the gateway queue manager.
   a. Create the queue managers.

      Run the command: `crtmqm -sax -u SYSTEM.DEAD.LETTER.QUEUE` *`QmgrName`* for each of the queue manager names in Table 4.

Table 4. Queue manager names and port numbers

Description	Queue Manager Name	Port number
Finance repository	`FINR1`	`1414`
Finance repository	`FINR2`	`1415`
Finance client	`FINCLT`	`1418`
Sales repository	`SALER1`	`1416`
Sales repository	`SALER2`	`1417`
Sales server	`SALESRV`	`1419`
Gateway	`GATE`	`1420`

   b. Start all the queue managers

      Run the command: `strmqm` *`QmgrName`* for each of the queue manager names in Table 4.

   c. Create the definitions for each of the queue managers

      Run the command: `runmqsc` *`QmgrName`* < *`filename`* where the files are listed in Figure 15 on page 49, and the file name matches the queue manager name.

finr1.txt
```
DEFINE LISTENER(1414) TRPTYPE(TCP) IPADDR(localhost) CONTROL(QMGR) PORT(1414) REPLACE
START LISTENER(1414)
ALTER QMGR REPOS(FINANCE)
DEFINE CHANNEL(FINANCE.FINR2) CHLTYPE(CLUSSDR) CONNAME('localhost(1415)') CLUSTER(FINANCE) REPLACE
DEFINE CHANNEL(FINANCE.FINR1) CHLTYPE(CLUSRCVR) CONNAME('localhost(1414)') CLUSTER(FINANCE) REPLACE
```

finr2.txt
```
DEFINE LISTENER(1415) TRPTYPE(TCP) IPADDR(localhost) CONTROL(QMGR) PORT(1415) REPLACE
START LISTENER(1415)
ALTER QMGR REPOS(FINANCE)
DEFINE CHANNEL(FINANCE.FINR1) CHLTYPE(CLUSSDR) CONNAME('localhost(1414)') CLUSTER(FINANCE) REPLACE
DEFINE CHANNEL(FINANCE.FINR2) CHLTYPE(CLUSRCVR) CONNAME('localhost(1415)') CLUSTER(FINANCE) REPLACE
```

finclt.txt
```
DEFINE LISTENER(1418) TRPTYPE(TCP) IPADDR(localhost) CONTROL(QMGR) PORT(1418) REPLACE
START LISTENER(1418)
DEFINE CHANNEL(FINANCE.FINR1) CHLTYPE(CLUSSDR)  CONNAME('localhost(1414)') CLUSTER(FINANCE) REPLACE
DEFINE CHANNEL(FINANCE.FINCLT) CHLTYPE(CLUSRCVR) CONNAME('localhost(1418)') CLUSTER(FINANCE) REPLACE
DEFINE QMODEL(SYSTEM.SAMPLE.REPLY) REPLACE
```

saler1.txt
```
DEFINE LISTENER(1416) TRPTYPE(TCP) IPADDR(localhost) CONTROL(QMGR) PORT(1416) REPLACE
START LISTENER(1416)
ALTER QMGR REPOS(SALES)
DEFINE CHANNEL(SALES.SALER2) CHLTYPE(CLUSSDR) CONNAME('localhost(1417)') CLUSTER(SALES) REPLACE
DEFINE CHANNEL(SALES.SALER1) CHLTYPE(CLUSRCVR) CONNAME('localhost(1416)') CLUSTER(SALES) REPLACE
```

saler2.txt
```
DEFINE LISTENER(1417) TRPTYPE(TCP) IPADDR(localhost) CONTROL(QMGR) PORT(1417) REPLACE
START LISTENER(1417)
ALTER QMGR REPOS(SALES)
DEFINE CHANNEL(SALES.SALER1) CHLTYPE(CLUSSDR) CONNAME('localhost(1416)') CLUSTER(SALES) REPLACE
DEFINE CHANNEL(SALES.SALER2) CHLTYPE(CLUSRCVR) CONNAME('localhost(1417)') CLUSTER(SALES) REPLACE
```

salesrv.txt
```
DEFINE LISTENER(1419) TRPTYPE(TCP) IPADDR(localhost) CONTROL(QMGR) PORT(1419) REPLACE
START LISTENER(1419)
DEFINE CHANNEL(SALES.SALER1) CHLTYPE(CLUSSDR) CONNAME('localhost(1416)') CLUSTER(SALES) REPLACE
DEFINE CHANNEL(SALES.SALESRV) CHLTYPE(CLUSRCVR) CONNAME('localhost(1419)') CLUSTER(SALES) REPLACE
DEFINE QLOCAL(SALES) CLUSTER(SALES) TRIGGER INITQ(SYSTEM.DEFAULT.INITIATION.QUEUE) PROCESS(ECHO) REPLACE
DEFINE PROCESS(ECHO) APPLICID(AMQSECH) REPLACE
```

gate.txt
```
DEFINE LISTENER(1420) TRPTYPE(TCP) IPADDR(LOCALHOST) CONTROL(QMGR) PORT(1420) REPLACE
START LISTENER(1420)
DEFINE NAMELIST(ALL) NAMES(SALES, FINANCE)
DEFINE CHANNEL(FINANCE.FINR1) CHLTYPE(CLUSSDR) CONNAME('LOCALHOST(1414)') CLUSTER(FINANCE) REPLACE
DEFINE CHANNEL(FINANCE.GATE) CHLTYPE(CLUSRCVR) CONNAME('LOCALHOST(1420)') CLUSTER(FINANCE) REPLACE
DEFINE CHANNEL(SALES.SALER1) CHLTYPE(CLUSSDR) CONNAME('LOCALHOST(1416)') CLUSTER(SALES) REPLACE
DEFINE CHANNEL(SALES.GATE)  CHLTYPE(CLUSRCVR) CONNAME('LOCALHOST(1420)') CLUSTER(SALES) REPLACE
DEFINE QALIAS(A.SALES) CLUSNL(ALL) TARGET(SALES) TARGTYPE(QUEUE) DEFBIND(NOTFIXED) REPLACE
DEFINE QREMOTE(FINCLT) RNAME(' ') RQMNAME(FINCLT) CLUSNL(ALL) REPLACE
DEFINE QREMOTE(SALESRV) RNAME(' ') RQMNAME(SALESRV) CLUSNL(ALL) REPLACE
```

Figure 15. Definitions for the basic clusters

2. Test the configuration by running the sample request program.
   a. Start the trigger monitor program on the `SALESRV` queue manager

      On Windows, open a command window and run the command `runmqtrm -m SALESRV`
   b. Run the sample request program, and send a request.

      On Windows, open a command window and run the command `amqsreq A.SALES FINCLT`

      The request message is echoed back, and after 15 seconds the sample program finishes.

3. Create the definitions to isolate the SALES queue in the Q.SALES cluster and separate cluster messages for the SALES and FINANCE cluster on the gateway queue manager.

   Run the command: `runmqsc` *QmgrName* < *filename* where the files are listed in Figure 16, and the file name almost matches the queue manager name.

chgsaler1.txt
```
DEFINE NAMELIST(CLUSTERS) NAMES(SALES, Q.SALES)
ALTER QMGR REPOS(' ') REPOSNL(CLUSTERS)
DEFINE CHANNEL(Q.SALES.SALER2) CHLTYPE(CLUSSDR) CONNAME('localhost(1417)') CLUSTER(Q.SALES) REPLACE
DEFINE CHANNEL(Q.SALES.SALER1) CHLTYPE(CLUSRCVR) CONNAME('localhost(1416)') CLUSTER(Q.SALES) REPLACE
```

chgsaler2.txt
```
DEFINE NAMELIST(CLUSTERS) NAMES(SALES, Q.SALES)
ALTER QMGR REPOS(' ') REPOSNL(CLUSTERS)
DEFINE CHANNEL(Q.SALES.SALER1) CHLTYPE(CLUSSDR) CONNAME('localhost(1416)') CLUSTER(Q.SALES) REPLACE
DEFINE CHANNEL(Q.SALES.SALER2) CHLTYPE(CLUSRCVR) CONNAME('localhost(1417)') CLUSTER(Q.SALES) REPLACE
```

chgsalesrv.txt
```
DEFINE NAMELIST (CLUSTERS) NAMES(SALES, Q.SALES)
DEFINE CHANNEL(Q.SALES.SALER1) CHLTYPE(CLUSSDR) CONNAME('localhost(1416)') CLUSTER(Q.SALES) REPLACE
DEFINE CHANNEL(Q.SALES.SAVESRV) CHLTYPE(CLUSRCVR) CONNAME('localhost(1419)') CLUSTER(Q.SALES) REPLACE
ALTER QLOCAL (SALES) CLUSTER(' ') CLUSNL(CLUSTERS)
```

chggate.txt
```
ALTER NAMELIST(ALL) NAMES(SALES, FINANCE, Q.SALES)
ALTER QMGR DEFCLXQ(CHANNEL)
DEFINE CHANNEL(Q.SALES.SALER1) CHLTYPE(CLUSSDR)  CONNAME('localhost(1416)') CLUSTER(Q.SALES) REPLACE
DEFINE CHANNEL(Q.SALES.GATE)  CHLTYPE(CLUSRCVR) CONNAME('localhost(1420)') CLUSTER(Q.SALES) REPLACE
DEFINE QLOCAL (XMITQ.Q.SALES.SALESRV) USAGE(XMITQ) CLCHNAME(Q.SALES.SALESRV) REPLACE
DEFINE QLOCAL (XMITQ.SALES)     USAGE(XMITQ) CLCHNAME(SALES.*)  REPLACE
DEFINE QLOCAL (XMITQ.FINANCE)    USAGE(XMITQ) CLCHNAME(FINANCE.*) REPLACE
```

Figure 16. Changes to isolate the sales queue in a new cluster and separate the gateway cluster transmission queues

4. Remove the SALES queue from the SALES cluster.

   Run the **MQSC** command in Figure 17:

```
ALTER QLOCAL(SALES) CLUSTER('Q.SALES') CLUSNL(' ')
```

Figure 17. Remove the sales queue on queue manager SALESRV from the sales cluster

5. Switch the channels to the new transmission queues.

   The requirement is to stop and start all the channels that the GATE queue manager is using. To do this with the least number of commands, stop and start the queue manager

   ```
   endmqm -i GATE
   strmqm GATE
   ```

What to do next

1. Rerun the sample request program to verify the new configuration works; see step 2 on page 49
2. Monitor the messages flowing through all the cluster transmission queues on the GATE queue manager:
   a. Alter the definition of each of the cluster transmission queues to turn queue monitoring on.

      `ALTER QLOCAL(SYSTEM.CLUSTER.TRANSMIT. *name*) STATQ(ON)`

   b. Check queue manager statistics monitoring is OFF, to minimize output, and set the monitoring interval to a lower value to perform multiple tests conveniently.

      ```
      ALTER QMGR STATINT(60) STATCHL(OFF) STATQ(OFF) STATMQI(OFF) STATACLS(OFF)
      ```

   c. Restart the GATE queue manager.

d. Run the sample request program a few times to verify that an equal number of messages are flowing through `SYSTEM.CLUSTER.TRANSMIT.Q.SALES.SALESRV` and `SYSTEM.CLUSTER.TRANSMIT.QUEUE`. Requests flow through `SYSTEM.CLUSTER.TRANSMIT.Q.SALES.SALESRV` and replies through `SYSTEM.CLUSTER.TRANSMIT.QUEUE`.

```
amqsmon -m GATE -t statistics
```

e. The results over a couple of intervals are as follows:

```
C:\Documents and Settings\Admin>amqsmon -m GATE -t statistics
MonitoringType: QueueStatistics
QueueManager: 'GATE'
IntervalStartDate: '2012-02-27'
IntervalStartTime: '14.59.20'
IntervalEndDate: '2012-02-27'
IntervalEndTime: '15.00.20'
CommandLevel: 700
ObjectCount: 2
QueueStatistics: 0
QueueName: 'SYSTEM.CLUSTER.TRANSMIT.QUEUE'
CreateDate: '2012-02-24'
CreateTime: '15.58.15'
...
Put1Count: [0, 0]
Put1FailCount: 0
PutBytes: [435, 0]
GetCount: [1, 0]
GetBytes: [435, 0]
...
QueueStatistics: 1
QueueName: 'SYSTEM.CLUSTER.TRANSMIT.Q.SALES.SAVESRV'
CreateDate: '2012-02-24'
CreateTime: '16.37.43'
...
PutCount: [1, 0]
PutFailCount: 0
Put1Count: [0, 0]
Put1FailCount: 0
PutBytes: [435, 0]
GetCount: [1, 0]
GetBytes: [435, 0]
...
MonitoringType: QueueStatistics
QueueManager: 'GATE'
IntervalStartDate: '2012-02-27'
IntervalStartTime: '15.00.20'
IntervalEndDate: '2012-02-27'
IntervalEndTime: '15.01.20'
CommandLevel: 700
ObjectCount: 2
QueueStatistics: 0
QueueName: 'SYSTEM.CLUSTER.TRANSMIT.QUEUE'
CreateDate: '2012-02-24'
CreateTime: '15.58.15'
...
PutCount: [2, 0]
PutFailCount: 0
Put1Count: [0, 0]
```

```
Put1FailCount: 0
PutBytes: [863, 0]
GetCount: [2, 0]
GetBytes: [863, 0]
...
QueueStatistics: 1
QueueName: 'SYSTEM.CLUSTER.TRANSMIT.Q.SALES.SAVESRV'
CreateDate: '2012-02-24'
CreateTime: '16.37.43'
...
PutCount: [2, 0]
PutFailCount: 0
Put1Count: [0, 0]
Put1FailCount: 0
PutBytes: [863, 0]
GetCount: [2, 0]
GetBytes: [863, 0]
...
2 Records Processed.
```

One request and reply message were sent in the first interval and two in the second. You can infer that the request messages were placed on `SYSTEM.CLUSTER.TRANSMIT.Q.SALES.SAVESRV`, and the reply messages on `SYSTEM.CLUSTER.TRANSMIT.QUEUE`.

Clustering: Switching cluster transmission queues:

Plan how the changes to the cluster transmission queues of an existing production queue manager are going to be brought into effect.

Before you begin

If you reduce the number of messages the switching process has to transfer to the new transmission queue, switching completes more quickly. Read How the process to switch cluster-sender channel to a different transmission queue works for the reasons for trying to empty the transmission queue before proceeding any further.

About this task

You have a choice of two ways of making the changes to cluster transmission queues take effect.

1. Let the queue manager make the changes automatically. This is the default. The queue manager switches cluster-sender channels with pending transmission queue changes when a cluster-sender channel next starts.
2. Make the changes manually. You can make the changes to a cluster-sender channel when it is stopped. You can switch it from one cluster transmission queue to another before the cluster-sender channel starts.

What factors do you take into account when deciding which of the two options to choose, and how do you manage the switch?

Procedure

- Option 1: Let the queue manager make the changes automatically; see “Switching active cluster-sender channels to another set of cluster-transmission queues” on page 54.

  Choose this option if you want the queue manager to make the switch for you.

  An alternative way to describe this option is to say the queue manager switches a cluster-sender channel without you forcing the channel to stop. You do have the option of forcing the channel to stop, and then starting the channel, to make the switch happen sooner. The switch starts when the channel starts, and runs while the channel is running, which is different to option 2. In option 2, the switch takes place when the channel is stopped.

If you choose this option by letting the switch happen automatically, the switching process starts when a cluster-sender channel starts. If the channel is not stopped, it starts after it becomes inactive, if there is a message to process. If the channel is stopped, start it with the `START CHANNEL` command.

The switch process completes as soon as there are no messages left for the cluster-sender channel on the transmission queue the channel was serving. As soon as that is the case, newly arrived messages for the cluster-sender channel are stored directly on the new transmission queue. Until then, messages are stored on the old transmission queue, and the switching process transfers messages from the old transmission queue to the new transmission queue. The cluster-sender channel forwards messages from the new cluster transmission queue during the whole switching process.

When the switch process completes depends on the state of the system. If you are making the changes in a maintenance window, assess beforehand whether the switching process will complete in time. Whether it will complete in time depends on whether the number of messages that are awaiting transfer from the old transmission queue reaches zero.

The advantage of the first method is it is automatic. A disadvantage is that if the time to make the configuration changes is limited to a maintenance window, you must be confident that you can control the system to complete the switch process inside the maintenance window. If you cannot be sure, option 2 might be a better choice.

- Option 2: Make the changes manually; see "Switching a stopped cluster-sender channel to another cluster transmission queue" on page 55.

Choose this option if you want to control the entire switching process manually, or if you want to switch a stopped or inactive channel. It is a good choice, if you are switching a few cluster-sender channels, and you want to do the switch during a maintenance window.

An alternative description of this option is to say that you switch the cluster-sender channel, while the cluster-sender channel is stopped.

If you choose this option you have complete control over when the switch takes place.

You can be certain about completing the switching process in a fixed amount of time, within a maintenance window. The time the switch takes depends on how many messages have to be transferred from one transmission queue to the other. If messages keep arriving, it might take a time for the process to transfer all the messages.

You have the option of switching the channel without transferring messages from the old transmission queue. The switch is "instant".

When you restart the cluster-sender channel, it starts processing messages on the transmission queue you newly assigned to it.

The advantage of the second method is you have control over the switching process. The disadvantage is that you must identify the cluster-sender channels to be switched, run the necessary commands, and resolve any in-doubt channels that might be preventing the cluster-sender channel stopping.

Related tasks:

"Switching active cluster-sender channels to another set of cluster-transmission queues"
This task gives you three options for switching active cluster-sender channels. One option is to let the queue manager make the switch automatically, which does not affect running applications. The other options are to stop and start channels manually, or to restart the queue manager.

"Switching a stopped cluster-sender channel to another cluster transmission queue" on page 55

Related information:

How the process to switch cluster-sender channel to a different transmission queue works

Switching active cluster-sender channels to another set of cluster-transmission queues:

This task gives you three options for switching active cluster-sender channels. One option is to let the queue manager make the switch automatically, which does not affect running applications. The other options are to stop and start channels manually, or to restart the queue manager.

Before you begin

Change the cluster transmission queue configuration. You can change the **DEFCLXQ** queue manager attribute, or add or modify the **CLCHNAME** attribute of transmission queues.

If you reduce the number of messages the switching process has to transfer to the new transmission queue, switching completes more quickly. Read How the process to switch cluster-sender channel to a different transmission queue works for the reasons for trying to empty the transmission queue before proceeding any further.

About this task

Use the steps in the task as a basis for working out your own plan for making cluster-transmission queue configuration changes.

Procedure

1. Optional: Record the current channel status

   Make a record of the status of current and saved channels that are serving cluster transmission queues. The following commands display the status associated with system cluster transmission queues. Add your own commands to display the status associated with cluster-transmission queues that you have defined. Use a convention, such as `XMITQ.` *`ChannelName`*, to name cluster transmission queues that you define to make it easy to display the channel status for those transmission queues.

   ```
   DISPLAY CHSTATUS(*) WHERE(XMITQ LK 'SYSTEM.CLUSTER.TRANSMIT.*')
   DISPLAY CHSTATUS(*) SAVED WHERE(XMITQ LK 'SYSTEM.CLUSTER.TRANSMIT.*')
   ```
2. Switch transmission queues.
   - Do nothing. The queue manager switches cluster-sender channels when they restart after being stopped or inactive.

     Choose this option if you have no rules or concerns about altering a queue manager configuration. Running applications are not affected by the changes.
   - Restart the queue manager. All cluster-sender channels are stopped and restarted automatically on demand.

     Choose this option to initiate all the changes immediately. Running applications are interrupted by the queue manager as it shuts down and restarts.
   - Stop individual cluster-sender channels and restart them.

     Choose this option to change a few channels immediately. Running applications experience a short delay in message transfer between your stopping and starting the message channel again. The cluster-sender channel remains running, except during the time you stopped it. During the switch process messages are delivered to the old transmission queue, transferred to the new transmission queue by the switching process, and forwarded from the new transmission queue by the cluster-sender channel.

3. Optional: Monitor the channels as they switch

   Display the channel status and the transmission queue depth during the switch. The following example display the status for system cluster transmission queues.

   ```
   DISPLAY CHSTATUS(*) WHERE(XMITQ LK 'SYSTEM.CLUSTER.TRANSMIT.*')
   DISPLAY CHSTATUS(*) SAVED WHERE(XMITQ LK 'SYSTEM.CLUSTER.TRANSMIT.*')
   DISPLAY QUEUE('SYSTEM.CLUSTER.TRANSMIT.*') CURDEPTH
   ```
4. Optional: Monitor the messages " AMQ7341 `The transmission queue for channel` *`ChannelName`* `switched from queue` *`QueueName`* `to` *`QueueName`* " that are written to the queue manager error log.

Switching a stopped cluster-sender channel to another cluster transmission queue:

Before you begin

You might make some configuration changes, and now want to make them effective without starting the cluster-sender channels that are affected. Alternatively, you make the configuration changes you require as one of the steps in the task.

If you reduce the number of messages the switching process has to transfer to the new transmission queue, switching completes more quickly. Read How the process to switch cluster-sender channel to a different transmission queue works for the reasons for trying to empty the transmission queue before proceeding any further.

About this task

This task switches the transmission queues served by stopped or inactive cluster-sender channels. You might do this task because a cluster-sender channel is stopped, and you want to switch its transmission queue immediately. For example, for some reason a cluster-sender channel is not starting, or has some other configuration problem. To resolve the problem, you decide to create a cluster-sender channel, and associate the transmission queue for the old cluster-sender channel with the new cluster-sender channel you defined.

A more likely scenario is you want to control when reconfiguration of cluster transmission queues is performed. To fully control the reconfiguration, you stop the channels, change the configuration, and then switch the transmission queues.

Procedure

1. Stop the channels that you intend to switch
   a. Stop any running or inactive channels that you intend to switch. Stopping an inactive cluster-sender channel prevents it starting while you are making configuration changes.

      ```
      STOP CHANNEL(ChannelName) MODE(QUIESCSE) STATUS(STOPPED)
      ```
2. Optional: Make the configuration changes.

   For example, see "Clustering: Example configuration of multiple cluster transmission queues" on page 45.
3. Switch the cluster-sender channels to the new cluster transmission queues. Multi On Multiplatforms, issue the following command:

   ```
   runswchl -m QmgrName -c ChannelName
   ```

   z/OS On z/OS, use the SWITCH function of the CSQUTIL command to switch the messages or monitor what is happening. Use the following command.

   ```
   SWITCH CHANNEL(channel_name) MOVEMSGS(YES)
   ```

For more information, see SWITCH function.

The `runswchl`, or CSQUTIL SWITCH, command transfers any messages on the old transmission queue to the new transmission queue. When the number of messages on the old transmission queue for this channel reaches zero, the switch is completed. The command is synchronous. The command writes progress messages to the window during the switching process.

During the transfer phase existing and new messages destined for the cluster-sender channel are transferred in order to the new transmission queue.

Because the cluster-sender channel is stopped, the messages build up on the new transmission queue. Contrast the stopped cluster-sender channel, to step 2 on page 54 in "Switching active cluster-sender channels to another set of cluster-transmission queues" on page 54. In that step, the cluster-sender channel is running, so messages do not necessarily build up on the new transmission queue.

4. Optional: Monitor the channels as they switch

   In a different command window, display the transmission queue depth during the switch. The following example display the status for system cluster transmission queues.

   ```
   DISPLAY QUEUE('SYSTEM.CLUSTER.TRANSMIT.*') CURDEPTH
   ```
5. Optional: Monitor the messages " AMQ7341 `The transmission queue for channel` *`ChannelName`* `switched from queue` *`QueueName`* `to` *`QueueName`* " that are written to the queue manager error log.
6. Restart the cluster-sender channels that you stopped.

   The channels do not start automatically, as you stopped them, placing them into `STOPPED` status.

   ```
   START CHANNEL(ChannelName)
   ```

Related information:

runswchl

RESOLVE CHANNEL

STOP CHANNEL

Clustering: Migration and modification best practices:

This topic provides guidance for planning and administering IBM MQ clusters. This information is a guide based on testing and feedback from customers.

1. "Moving objects in a cluster" (Best practices for moving objects around inside a cluster, without installing any fix packs or new versions of IBM MQ).
2. "Upgrades and maintenance installations" on page 57 (Best practices for keeping a working cluster architecture up and running, while applying maintenance or upgrades and testing the new architecture).

Moving objects in a cluster

Applications and their queues

When you must move a queue instance hosted on one queue manager to be hosted on another, you can work with the workload balancing parameters to ensure a smooth transition.

Create an instance of the queue where it is to be newly hosted, but use cluster workload balancing settings to continue sending messages to the original instance until your application is ready to switch. This is achieved with the following steps:

1. Set the **`CLWLRANK`** property of the existing queue to a high value, for example five.
2. Create the new instance of the queue and set its **`CLWLRANK`** property to zero.
3. Complete any further configuration of the new system, for example deploy and start consuming applications against the new instance of the queue.
4. Set the **`CLWLRANK`** property of the new queue instance to be higher than the original instance, for example nine.

5. Allow the original queue instance to process any queued messages in the system and then delete the queue.

Moving entire queue managers

If the queue manager is staying on the same host, but the IP address is changing, then the process is as follows:

- DNS, when used correctly, can help simplify the process. For information about using DNS by setting the Connection name (CONNAME) channel attribute, see ALTER CHANNEL.
- If moving a full repository, ensure that you have at least one other full repository which is running smoothly (no problems with channel status for example) before making changes.
- Suspend the queue manager using the SUSPEND QMGR command to avoid traffic buildup.
- Modify the IP address of the computer. If your CLUSRCVR channel definition uses an IP address in the CONNAME field, modify this IP address entry. The DNS cache might need to be flushed through to ensure that updates are available everywhere.
- When the queue manager reconnects to the full repositories, channel auto-definitions automatically resolve themselves.
- If the queue manager hosted a full repository and the IP address changes, it is important to ensure that partials are switched over as soon as possible to point any manually defined CLUSSDR channels to the new location. Until this switch is carried out, these queue managers might be able to only contact the remaining (unchanged) full repository, and warning messages might be seen regarding the incorrect channel definition.
- Resume the queue manager using the RESUME QMGR command.

If the queue manager must be moved to a new host, it is possible to copy the queue manager data and restore from a backup. This process is not recommended however, unless there are no other options; it might be better to create a queue manager on a new machine and replicate queues and applications as described in the previous section. This situation gives a smooth rollover/rollback mechanism.

If you are determined to move a complete queue manager using backup, follow these best practices:

- Treat the whole process as a queue manager restore from backup, applying any processes you would usually use for system recovery as appropriate for your operating system environment.
- Use the `REFRESH CLUSTER` command after migration to discard all locally held cluster information (including any auto-defined channels that are in doubt), and force it to be rebuilt.

  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.

When creating a queue manager and replicating the setup from an existing queue manager in the cluster (as described previously in this topic), never treat the two different queue managers as actually being the same. In particular, do not give a new queue manager the same queue manager name and IP address. Attempting to 'drop in' a replacement queue manager is a frequent cause of problems in IBM MQ clusters. The cache expects to receive updates including the `QMID` attribute, and state can be corrupted.

If two different queue managers are accidentally created with the same name, it is recommended to use the RESET CLUSTER `QMID` command to eject the incorrect entry from the cluster.

Upgrades and maintenance installations

Avoid the so-called big bang scenario (for example, stopping all cluster and queue manager activity, applying all upgrades and maintenance to all queue managers, then starting everything at the same time). Clusters are designed to still work with multiple versions of queue manager coexisting, so a well-planned, phased maintenance approach is recommended.

Have a backup plan:

- z/OS On z/OS, have you applied backwards migration PTFs?
- Have you taken backups?
- Avoid using new cluster functionality immediately: Wait until you are sure that all the queue managers are upgraded to the new level, and are certain that you are not going to roll any of them back. Using new cluster function in a cluster where some queue managers are still at an earlier level can lead to undefined behavior. For example, in the move to IBM WebSphere® MQ Version 7.1 from IBM WebSphere MQ Version 6.0, if a queue manager defines a cluster topic, IBM WebSphere MQ Version 6.0 queue managers will not understand the definition or be able to publish on this topic.

Migrate full repositories first. Although they can forward information that they do not understand, they cannot persist it, so it is not the recommended approach unless absolutely necessary. For more information, see Queue manager cluster migration.

Clustering: Using REFRESH CLUSTER best practices:

You use the **REFRESH CLUSTER** command to discard all locally held information about a cluster and rebuild that information from the full repositories in the cluster. You should not need to use this command, except in exceptional circumstances. If you do need to use it, there are special considerations about how you use it. This information is a guide based on testing and feedback from customers.

Only run REFRESH CLUSTER if you really need to do so

The IBM MQ cluster technology ensures that any change to the cluster configuration, such as a change to a clustered queue, automatically becomes known to any member of the cluster that needs to know the information. There is no need for further administrative steps to be taken to achieve this propagation of information.

If such information does not reach the queue managers in the cluster where it is required, for example a clustered queue is not known by another queue manager in the cluster when an application attempts to open it for the first time, it implies a problem in the cluster infrastructure. For example, it is possible that a channel cannot be started between a queue manager and a full repository queue manager. Therefore, any situation where inconsistencies are observed must be investigated. If possible, resolve the situation without using the **REFRESH CLUSTER** command.

In rare circumstances that are documented elsewhere in this product documentation, or when requested by IBM support, you can use the **REFRESH CLUSTER** command to discard all locally held information about a cluster and rebuild that information from the full repositories in the cluster.

Refreshing in a large cluster can affect performance and availability of the cluster

Use of the **REFRESH CLUSTER** command can be disruptive to the cluster while it is in progress, for example by creating a sudden increase in work for the full repositories as they process the repropagation of queue manager cluster resources. If you are refreshing in a large cluster (that is, many hundreds of queue managers) you should avoid use of the command in day-to-day work if possible and use alternative methods to correct specific inconsistencies. For example, if a cluster queue is not being correctly propagated across the cluster, an initial investigation technique of updating the clustered queue definition, such as altering its description, repropagates the queue configuration across the cluster. This process can help to identify the problem and potentially resolve a temporary inconsistency.

If alternative methods cannot be used, and you have to run **REFRESH CLUSTER** in a large cluster, you should do so at off-peak times or during a maintenance window to avoid impact on user workloads. You should also avoid refreshing a large cluster in a single batch, and instead stagger the activity as explained in "Avoid performance and availability issues when cluster objects send automatic updates."

Avoid performance and availability issues when cluster objects send automatic updates

After a new cluster object is defined on a queue manager, an update for this object is generated every 27 days from the time of definition, and sent to every full repository in the cluster and onwards to any other interested queue managers. When you issue the **`REFRESH CLUSTER`** command to a queue manager, you reset the clock for this automatic update on all objects defined locally in the specified cluster.

If you refresh a large cluster (that is, many hundreds of queue managers) in a single batch, or in other circumstances such as recreating a system from configuration backup, after 27 days all of those queue managers will re-advertise all of their object definitions to the full repositories at the same time. This could again cause the system to run significantly slower, or even become unavailable, until all the updates have completed. Therefore, when you have to refresh or re-create multiple queue managers in a large cluster, you should stagger the activity over several hours, or several days, so that subsequent automatic updates do not regularly impact system performance.

The system cluster history queue

When a **`REFRESH CLUSTER`** is performed, the queue manager takes a snapshot of the cluster state before the refresh and stores it on the `SYSTEM.CLUSTER.HISTORY.QUEUE` (SCHQ) if it is defined on the queue manager. This snapshot is for IBM service purposes only, in case of later problems with the system.

The SCHQ is defined by default on distributed queue managers on startup. For z/OS migration, the SCHQ must be manually defined.

Messages on the SCHQ expire after three months.

Related information:

REFRESH CLUSTER considerations for publish/subscribe clusters

Clustering: Availability, multi-instance, and disaster recovery:

This topic provides guidance for planning and administering IBM MQ clusters. This information is a guide based on testing and feedback from customers.

IBM MQ Clustering itself is not a High Availability solution, but in some circumstances it can be used to improve the availability of services using IBM MQ, for example by having multiple instances of a queue on different queue managers. This section gives guidance on ensuring that the IBM MQ infrastructure is as highly available as possible so that it can be used in such an architecture.

Availability of cluster resources

The reason for the usual recommendation to maintain two full repositories is that the loss of one is not critical to the smooth running of the cluster. Even if both become unavailable, there is a 60 day grace period for existing knowledge held by partial repositories, although new or not previously accessed resources (queues for example) are not available in this event.

Using clusters to improve application availability

A cluster can help in designing highly available applications (for example a request/response type server application), by using multiple instances of the queue and application. If needed, priority attributes can give preference to the 'live' application unless a queue manager or channel for example become unavailable. This is powerful for switching over quickly to continue processing new messages when a problem occurs.

However, messages which were delivered to a particular queue manager in a cluster are held only on that queue instance, and are not available for processing until that queue manager is recovered. For this reason, for true data high availability you might want to consider other technologies such as multi-instance queue managers.

Multi-instance queue managers

Software High Availability (multi-instance) is the best built-in offering for keeping your existing messages available. See Using IBM MQ with high availability configurations, Create a multi-instance queue manager, and the following section for more information. Any queue manager in a cluster may be made highly available using this technique, as long as all queue managers in the cluster are running at least IBM WebSphere MQ Version 7.0.1. If any queue managers in the cluster are at previous levels, they might lose connectivity with the multi-instance queue managers if they fail over to a secondary IP.

As discussed previously in this topic, as long as two full repositories are configured, they are almost by their nature highly available. If you need to, IBM MQ software High Availability / multi-instance queue managers can be used for full repositories. There is no strong reason to use these methods, and in fact for temporary outages these methods might cause additional performance cost during the failover. Using software HA instead of running two full repositories is discouraged because in the event of a single channel outage, for example, it would not necessarily fail over, but might leave partial repositories unable to query for cluster resources.

Disaster recovery

Disaster recovery, for example recovering from when the disks storing a queue manager's data becomes corrupt, is difficult to do well; IBM MQ can help, but it cannot do it automatically. The only 'true' disaster recovery option in IBM MQ (excluding any operating system or other underlying replication technologies) is restoration from a backup. There are some cluster specific points to consider in these situations:

- Take care when testing disaster recovery scenarios. For example, if testing the operation of backup queue managers, be careful when bringing them online in the same network as it is possible to accidentally join the live cluster and start 'stealing' messages by hosting the same named queues as in the live cluster queue managers.
- Disaster recovery testing must not interfere with a running live cluster. Techniques to avoid interference include:
  - Complete network separation or separation at the firewall level.
  - z/OS Not starting channel initiation or the z/OS **chinit** address space.
  - Not issuing live TLS certificate to the disaster recovery system until, or unless, an actual disaster recovery scenario occurs.
- When restoring a backup of a queue manager in the cluster it is possible that the backup is out of sync with the rest of the cluster. The **REFRESH CLUSTER** command can resolve updates and synchronize with the cluster but the **REFRESH CLUSTER** command must be used as a last resort. See “Clustering: Using REFRESH CLUSTER best practices” on page 58. Review any in-house process documentation and IBM MQ documentation to see whether a simple step was missed before resorting to using the command.
- As for any recovery, applications must deal with replay and loss of data. It must be decided whether to clear the queues down to a known state, or if there is enough information elsewhere to manage replays.

Planning your distributed publish/subscribe network

You can create a network of queue managers where subscriptions created on one queue manager will receive matching messages published by an application connected to another queue manager in the network. To choose a suitable topology, you need to consider your requirements for manual control, network size, frequency of change, availability and scalability.

Before you begin

This task assumes that you understand what distributed publish/subscribe networks are, and how they work. For a technical overview, see Distributed publish/subscribe networks.

About this task

There are three basic topologies for a publish/subscribe network:
- Direct routed cluster
- Topic host routed cluster
- Hierarchy

For the first two topologies, the starting point is an IBM MQ cluster configuration. The third topology can be created with or without a cluster. See"Planning your distributed queues and clusters" on page 7for information about planning the underlying queue manager network.

A *Direct routed cluster* is the simplest topology to configure when a cluster is already present. Any topic that you define on any queue manager is automatically made available on every queue manager in the cluster, and publications are routed directly from any queue manager where a publishing application connects, to each of the queue managers where matching subscriptions exist. This simplicity of configuration relies on IBM MQ maintaining a high level of sharing of information and connectivity between every queue manager in the cluster. For small and simple networks (that is, a small number of queue managers, and a fairly static set of publishers and subscribers) this is acceptable. However, when used in larger or more dynamic environments the overhead might be prohibitive. See"Direct routing in publish/subscribe clusters" on page 66.

A *Topic host routed cluster* gives the same benefit as a direct routed cluster, by making any topic that you define on any queue manager in the cluster automatically available on every queue manager in the cluster. However, topic host routed clusters require you to carefully choose the queue managers that host each topic, because all information and publications for that topic pass through those topic host queue managers. This means that the system does not have to maintain channels and information flows between all queue managers. However it also means that publications might no longer be sent directly to subscribers, but might be routed through a topic host queue manager. For these reasons additional load might be put on the system, especially on the queue managers hosting the topics, so careful planning of the topology is required. This topology is particularly effective for networks that contain many queue managers, or that host a dynamic set of publishers and subscribers (that is, publishers or subscribers that are frequently added or removed). Additional topic hosts can be defined to improve availability of routes and to horizontally scale publication workload. See"Topic host routing in publish/subscribe clusters" on page 71.

A *Hierarchy* requires the most manual configuration to set up, and is the hardest topology to modify. You must manually configure the relationships between each queue manager in the hierarchy and its direct relations. After relationships are configured, publications will (as for the previous two topologies) be routed to subscriptions on other queue managers in the hierarchy. Publications are routed using the hierarchy relationships. This allows very specific topologies to be configured to suit different requirements, but it can also result in publications requiring many "hops" through intermediate queue managers to reach the subscriptions. There is always only one route through a hierarchy for a publication, so availability of every queue manager is critical. Hierarchies are typically only preferable where a single cluster cannot be configured; for example when spanning multiple organizations. See“Routing in publish/subscribe hierarchies” on page 96.

Where necessary, the above three topologies can be combined to solve specific topographical requirements. For an example, see Combining the topic spaces of multiple clusters.

To choose a suitable topology for your distributed publish/subscribe network, you need to consider the following broad questions:
- How big will your network be?
- How much manual control do you need over its configuration?
- How dynamic will the system be, both in terms of topics and subscriptions, and in terms of queue managers?
- What are your availability and scalability requirements?
- Can all queue managers connect directly to each other?

Procedure

- Estimate how big your network needs to be.
  1. Estimate how many topics you need.
  2. Estimate how many publishers and subscribers you expect to have.
  3. Estimate how many queue managers will be involved in publish/subscribe activities.

     See also“Publish/subscribe clustering: Best practices” on page 81, especially the following sections:
     – How to size your system
     – Reasons to limit the number of cluster queue managers involved in publish/subscribe activity
     – How to decide which topics to cluster

  If your network will have many queue managers, and handle many publishers and subscribers, you probably need to use a topic host routed cluster or a hierarchy. Direct routed clusters require almost no manual configuration, and can be a good solution for small or static networks.
- Consider how much manual control you need over which queue manager hosts each topic, publisher or subscriber.
  1. Consider whether some of your queue managers are less capable than others.
  2. Consider whether the communication links to some of your queue managers are more fragile than to others.
  3. Identify cases where you expect a topic to have many publications and few subscribers.
  4. Identify cases where you expect a topic to have many subscribers and few publications.

  In all topologies, publications are delivered to subscriptions on other queue managers. In a direct routed cluster those publications take the shortest path to the subscriptions. In a topic host routed cluster or a hierarchy, you control the route that publications take. If your queue managers differ in their capability, or have differing levels of availability and connectivity, you will probably want to assign specific workloads to specific queue managers. You can do this using either a topic host routed cluster or a hierarchy.

  In all topologies, co-locating the publishing applications on the same queue manager as the subscriptions whenever possible minimizes overheads and maximizes performance. For topic host routed clusters, consider putting publishers or subscribers on the queue managers that host the topic. This removes any extra “hops” between queue managers to pass a publication to a subscriber. This approach is particularly effective in cases where a topic has many publishers and few subscribers, or many subscribers and few publishers. See, for example, Topic host routing using centralized publishers or subscribers.

  See also“Publish/subscribe clustering: Best practices” on page 81, especially the following sections:
  – How to decide which topics to cluster
  – Publisher and subscription location

- Consider how dynamic the network activity will be.
  1. Estimate how frequently subscribers will be added and removed on different topics.

     Whenever a subscription is added or removed from a queue manager, and it is the first or last subscription for that specific topic string, that information is communicated to other queue managers in the topology. In a direct routed cluster and a hierarchy, this subscription information is propagated to every queue manager in the topology whether or not they have publishers on the topic. If the topology consists of many queue managers, this might be a significant performance overhead. In a topic host routed cluster, this information is only propagated to those queue managers that host a clustered topic that maps to the subscription's topic string.

     See also the Subscription change and dynamic topic strings section of"Publish/subscribe clustering: Best practices" on page 81.

     Note: In very dynamic systems, where the set of many unique topic strings is rapidly and constantly being changed, it might be best to switch the model to a "publish everywhere" mode. See Subscription performance in publish/subscribe networks.
  2. Consider how dynamic the queue managers are in the topology.

     A hierarchy requires each change in queue manager in the topology to be manually inserted or removed from the hierarchy, with care taken when changing queue managers at higher levels in the hierarchy. Queue managers in a hierarchy typically also use manually configured channel connections. You must maintain these connections, adding and removing channels as queue managers are added and removed from the hierarchy.

     In a publish/subscribe cluster, queue managers are automatically connected to any other queue manager that is required when they first join the cluster, and automatically become aware of topics and subscriptions.
- Consider your route availability and publication traffic scalability requirements.
  1. Decide whether you need to always have an available route from a publishing queue manager to a subscribing queue manager, even when a queue manager is unavailable.
  2. Consider how scalable you need the network to be. Decide whether the level of publication traffic is too high to be routed through a single queue manager or channel, and whether that level of publication traffic must be handled by a single topic branch or can be spread across multiple topic branches.
  3. Consider whether you need to maintain message ordering.

  Because a direct routed cluster sends messages directly from publishing queue managers to subscribing queue managers, you do not need to consider the availability of intermediate queue managers along the route. Similarly, scaling to the intermediate queue managers is not a consideration. However, as previously mentioned, the overhead of automatically maintaining channels and information flows between all queue managers in the cluster can significantly affect performance, especially in a large or dynamic environment.

  A topic host routed cluster can be tuned for individual topics. You can ensure that each branch of the topic tree that has a considerable publication workload is defined on a different queue manager, and that each queue manager is sufficiently performant and available for the expected workload for that branch of the topic tree. You can also improve availability and horizontal scaling further by defining each topic on multiple queue managers. This allows the system to route around unavailable topic host queue managers, and to workload balance publication traffic across them. However, when you define a given topic on multiple queue managers, you also introduce the following constraints:
  - You lose message ordering across publications.
  - You cannot use retained publications. See"Design considerations for retained publications in publish/subscribe clusters" on page 94.

  You cannot configure high availability or scalability of routing in a hierarchy through multiple routes.

  See also the Publication traffic section of"Publish/subscribe clustering: Best practices" on page 81.
- Based on these calculations, use the links provided to help you decide whether to use a topic host routed cluster, a direct routed cluster, a hierarchy, or a mixture of these topologies.

What to do next

You are now ready to configure your distributed publish/subscribe network.

Related information:

Configuring a queue manager cluster

Configuring distributed queuing

Configuring a publish/subscribe cluster

Connecting a queue manager to a publish/subscribe hierarchy

Designing publish/subscribe clusters

There are two basic publish/subscribe cluster topologies: *direct routing* and *topic host routing*. Each has different benefits. When you design your publish/subscribe cluster, choose the topology that best fits your expected network requirements.

For an overview of the two publish/subscribe cluster topologies, see Publish/subscribe clusters. To help you evaluate your network requirements, see “Planning your distributed publish/subscribe network” on page 61 and “Publish/subscribe clustering: Best practices” on page 81.

In general, both cluster topologies provide the following benefits:

- Simple configuration on top of a point-to-point cluster topology.
- Automatic handling of queue managers joining and leaving the cluster.
- Ease of scaling for additional subscriptions and publishers, by adding extra queue managers and distributing the additional subscriptions and publishers across them.

However, the two topologies have different benefits as the requirements become more specific.

Direct routed publish/subscribe clusters

With direct routing, any queue manager in the cluster sends publications from connected applications direct to any other queue manager in the cluster with a matching subscription.

A direct routed publish/subscribe cluster provides the following benefits:

- Messages destined for a subscription on a specific queue manager in the same cluster are transported directly to that queue manager and do not need to pass through an intermediate queue manager. This can improve performance in comparison with a topic host routed topology, or a hierarchical topology.
- Because all queue managers are directly connected to each other, there is no single point of failure in the routing infrastructure of this topology. If one queue manager is not available, subscriptions on other queue managers in the cluster are still able to receive messages from publishers on available queue managers.
- It is very simple to configure, especially on an existing cluster.

Things to consider when using a direct routed publish/subscribe cluster:

- All queue managers in the cluster become aware of all other queue managers in the cluster.
- Queue managers in a cluster that host one or more subscriptions to a clustered topic, automatically create cluster sender channels to all other queue managers in the cluster, even when those queue managers are not publishing messages on any clustered topics.
- The first subscription on a queue manager to a topic string under a clustered topic results in a message being sent to every other queue manager in the cluster. Similarly, the last subscription on a topic string to be deleted also results in a message. The more individual topic strings being used under a clustered topic, and the higher the rate of change of subscriptions, the more inter-queue manager communication occurs.

- Every queue manager in the cluster maintains the knowledge of subscribed topic strings that it is informed of, even when the queue manager is neither publishing nor subscribing to those topics.

For the above reasons, all queue managers in a cluster with a direct routed topic defined will incur an additional overhead. The more queue managers there are in the cluster, the greater the overhead. Likewise the more topic strings subscribed to, and the greater their rate of change, the greater the overhead. This can result in too much load on queue managers running on small systems in a large or dynamic direct routed publish/subscribe cluster. See Direct routed publish/subscribe performance for further information.

When you know that a cluster cannot accommodate the overheads of direct routed clustered publish/subscribe, you can instead use topic host routed publish/subscribe. Alternatively, in extreme situations, you can completely disable clustered publish/subscribe functionality by setting the queue manager attribute **PSCLUS** to `DISABLED` on every queue manager in the cluster. See "Inhibiting clustered publish/subscribe" on page 91. This prevents any clustered topic from being created, and therefore ensures that your network does not incur any overheads associated with clustered publish/subscribe.

Topic host routed publish/subscribe clusters

With topic host routing, the queue managers where clustered topics are administratively defined become routers for publications. Publications from non-hosting queue managers in the cluster are routed through the hosting queue manager to any queue manager in the cluster with a matching subscription.

A topic host routed publish/subscribe cluster provides the following extra benefits over a direct routed publish/subscribe cluster:
- Only queue managers on which topic host routed topics are defined are made aware of all other queue managers in the cluster.
- Only the topic host queue managers need to be able to connect to all other queue managers in the cluster, and will typically only connect to those where subscriptions exist. Therefore there are significantly fewer channels running between queue managers.
- Cluster queue managers that host one or more subscriptions to a clustered topic automatically create cluster sender channels only to queue managers that host a cluster topic that maps to the topic string of the subscription.
- The first subscription on a queue manager to a topic string under a clustered topic results in a message being sent to a queue manager in the cluster that hosts the clustered topic. Similarly, the last subscription on a topic string to be deleted also results in a message. The more individual topic strings being used under a clustered topic, and the higher the rate of change of subscriptions, the more inter-queue manager communication occurs, but only between subscription hosts and topic hosts.
- More control over the physical configuration. With direct routing, all queue managers have to participate in the publish/subscribe cluster, increasing their overheads. With topic host routing, only the topic host queue managers are aware of other queue managers and their subscriptions. You explicitly choose the topic host queue managers, therefore you can ensure that those queue managers are running on adequate equipment, and you can use less powerful systems for the other queue managers.

Things to consider when using a topic host routed publish/subscribe cluster:
- An extra "hop" between a publishing queue manager and a subscribing queue manager is introduced when the publisher or the subscriber is not located on a topic hosting queue manager. The latency caused by the extra "hop" can mean that topic host routing is less efficient that direct routing.
- On large clusters, topic host routing eases the significant performance and scaling issues that you can get with direct routing.
- You might choose to define all your topics on a single queue manager, or on a very small number of queue managers. If you do this, make sure the topic host queue managers are hosted on powerful systems with good connectivity.

- You can define the same topic on more than one queue manager. This improves the availability of the topic, and also improves scalability because IBM MQ workload balances publications for a topic across all hosts for that topic. Note, however, that defining the same topic on more than one queue manager loses message order for that topic.
- By hosting different topics on different queue managers, you can improve scalability without losing message order.

Related information:

Publish/subscribe cluster scenario

Configuring a publish/subscribe cluster

Tuning distributed publish/subscribe networks

Distributed publish/subscribe troubleshooting

Direct routing in publish/subscribe clusters:

Publications from any publishing queue manager are routed direct to any other queue manager in the cluster with a matching subscription.

For an introduction to how messages are routed between queue managers in publish/subscribe hierarchies and clusters, see Distributed publish/subscribe networks.

A direct routed publish/subscribe cluster behaves as follows:

- All queue managers automatically know of all other queue managers.
- All queue managers with subscriptions to clustered topics create channels to all other queue managers in the cluster and inform them of their subscriptions.
- Messages published by an application are routed from the queue manager that it is connected to, direct to each queue manager where a matching subscription exists.

The following diagram shows a queue manager cluster that is not currently used for publish/subscribe or point-to-point activities. Note that every queue manager in the cluster connects only to and from the full repository queue managers.

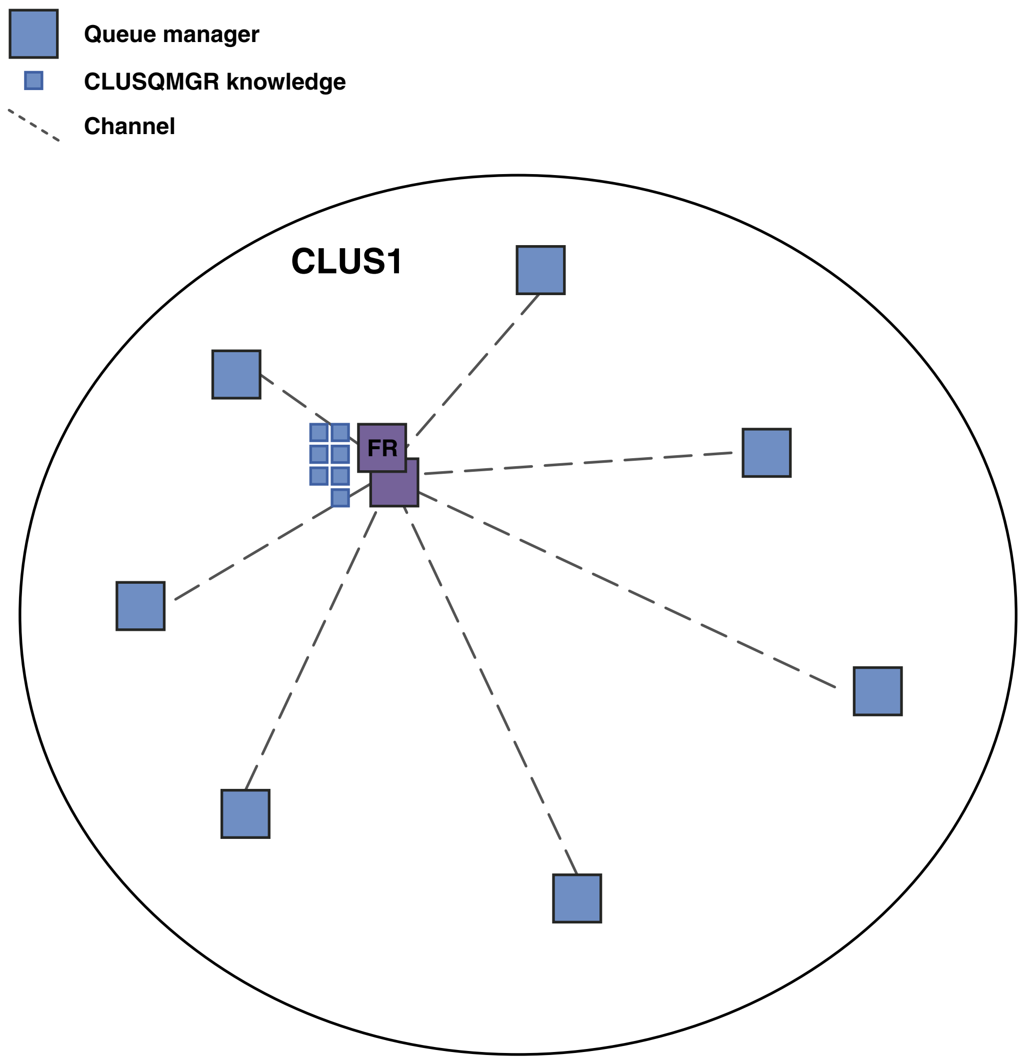


Figure 18. A queue manager cluster

For publications to flow between queue managers in a direct routed cluster, you cluster a branch of the topic tree as described in Configuring a publish/subscribe cluster, and specify *direct routing* (the default).

In a direct routed publish/subscribe cluster, you define the topic object on any queue manager in the cluster. When you do this, knowledge of the object, and knowledge of all other queue managers in the cluster, is automatically pushed to all queue managers in the cluster by the full repository queue managers. This happens before any queue manager references the topic:

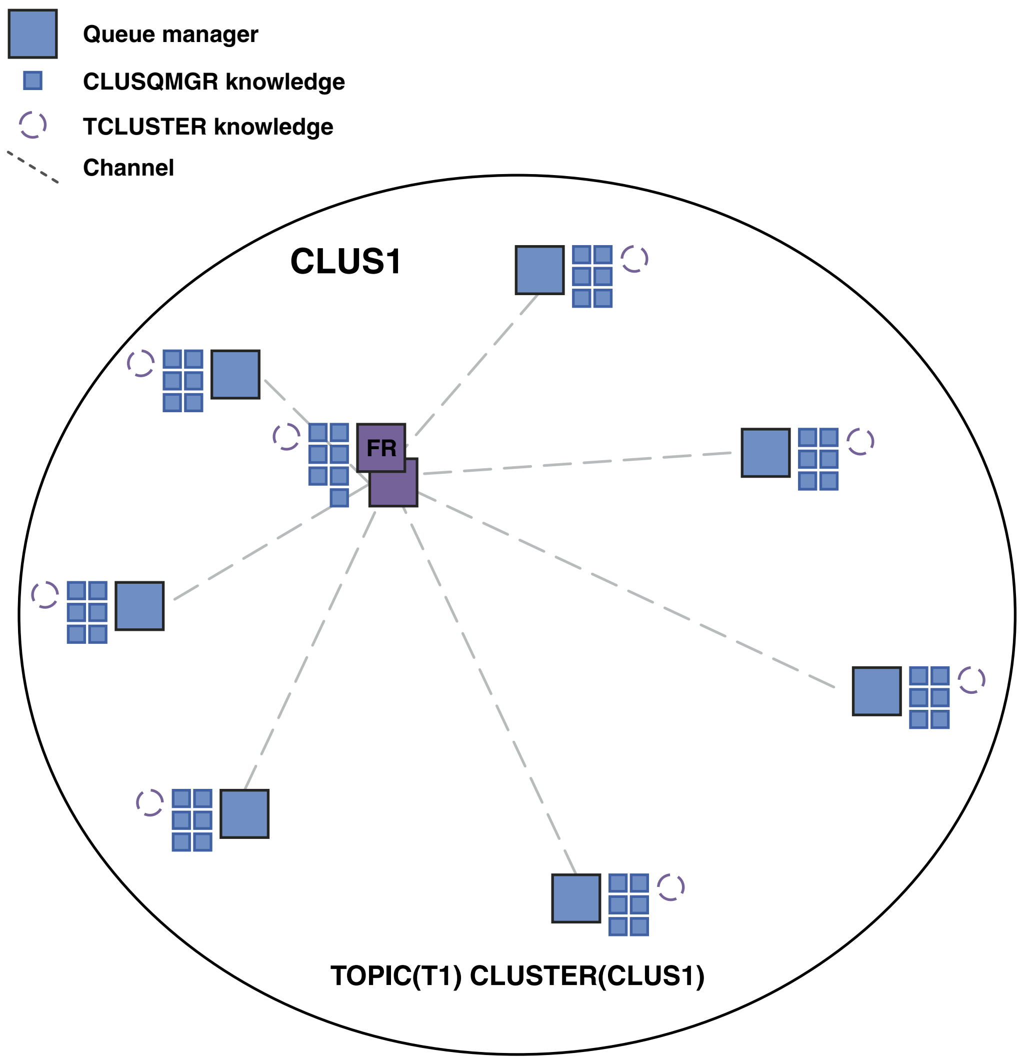


Figure 19. A direct routed publish/subscribe cluster

When a subscription is created, the queue manager that hosts the subscription establishes a channel to every queue manager in the cluster, and sends details of the subscription. This distributed subscription knowledge is represented by a proxy subscription on each queue manager. When a publication is produced on any queue manager in the cluster that matches that proxy subscription's topic string, a cluster channel is established from the publisher queue manager to each queue manager hosting a subscription, and the message is sent to each of them.

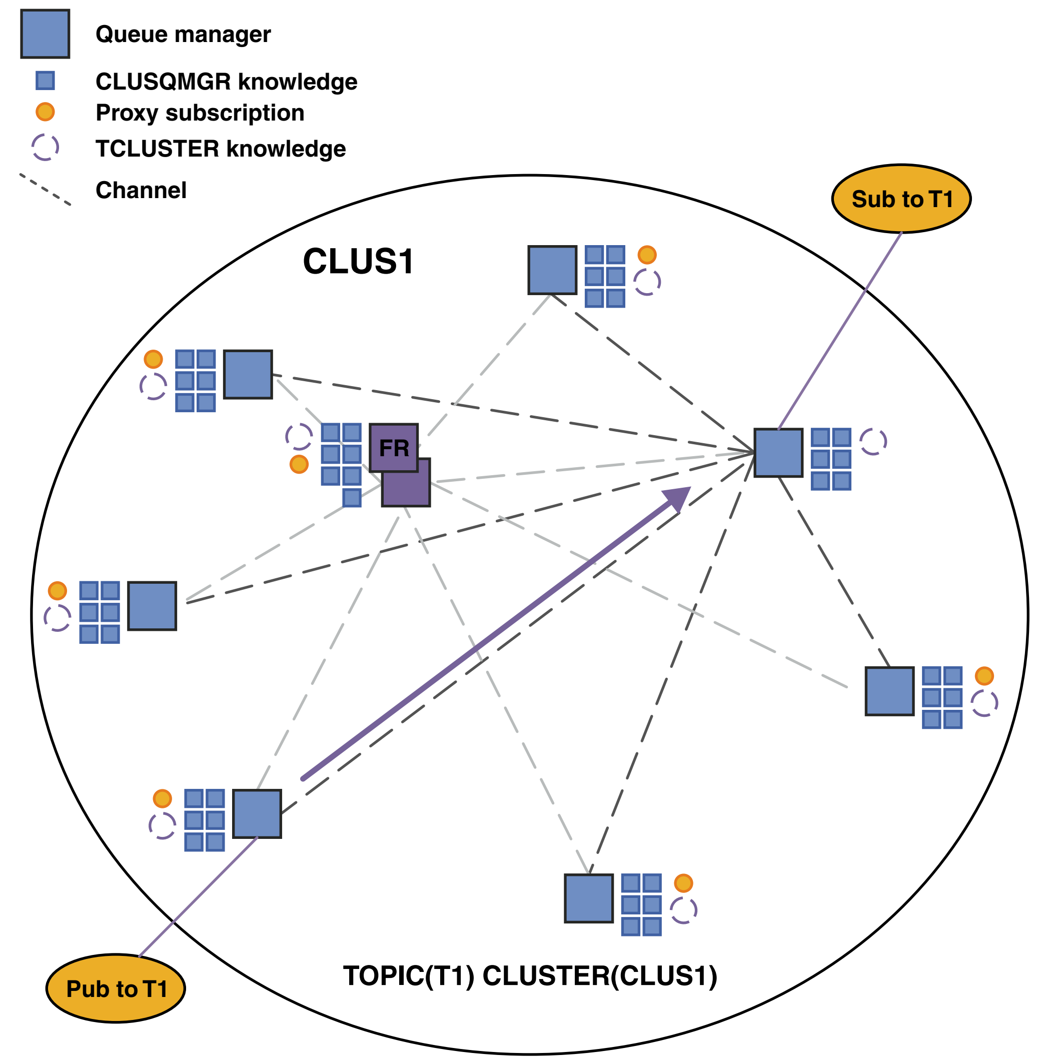


Figure 20. A direct routed publish/subscribe cluster with a publisher and a subscriber to a clustered topic

The direct routing of publications to subscription hosting queue managers simplifies configuration and minimizes the latency in delivering publications to subscriptions.

However, depending on the location of subscriptions and publishers, your cluster can quickly become fully interconnected, with every queue manager having a direct connection to every other queue manager. This might or might not be acceptable in your environment. Similarly, if the set of topic strings being subscribed to is changing frequently, the overhead of propagating that information between all queue managers can also become significant. All queue managers in a direct routed publish/subscribe cluster must be able to cope with these overheads.

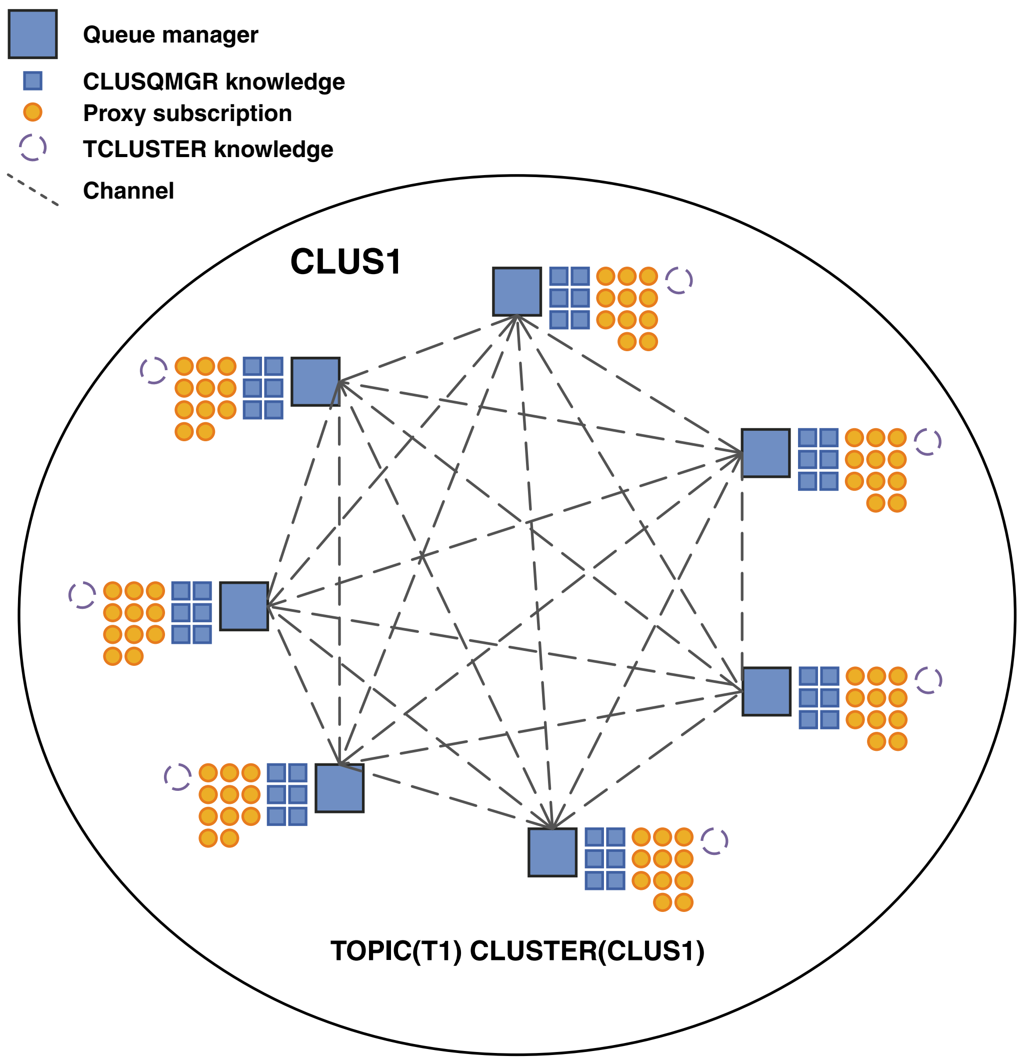


Figure 21. A direct routed publish/subscribe cluster that is fully interconnected

Summary and additional considerations

A direct routed publish/subscribe cluster needs little manual intervention to create or administer, and provides direct routing between publishers and subscribers. For certain configurations it is usually the most appropriate topology, notably clusters with few queue managers, or where high queue manager connectivity is acceptable and subscriptions are changing infrequently. However it also imposes certain constraints upon your system:

- The load on each queue manager is proportional to the total number of queue managers in the cluster. Therefore, in larger clusters, individual queue managers and the system as a whole can experience performance issues.
- By default, all clustered topic strings subscribed to are propagated throughout the cluster, and publications are propagated only to remote queue managers that have a subscription to the associated topic. Therefore rapid changes to the set of subscriptions can become a limiting factor. You can change this default behavior, and instead have all publications propagated to all queue managers, which removes the need for proxy subscriptions. This reduces the subscription knowledge traffic, but is likely to increase the publication traffic and the number of channels each queue manager establishes. See Subscription performance in publish/subscribe networks.

  Note: A similar restriction also applies to hierarchies.

- Because of the interconnected nature of publish/subscribe queue managers, it takes time for proxy subscriptions to propagate around all nodes in the network. Remote publications do not necessarily start being subscribed to immediately, so early publications might not be sent following a subscription to a new topic string. You can remove the problems caused by the subscription delay by having all publications propagated to all queue managers, which removes the need for proxy subscriptions. See Subscription performance in publish/subscribe networks.

  Note: This restriction also applies to hierarchies.

Before you use direct routing, explore the alternative approaches detailed in "Topic host routing in publish/subscribe clusters," and "Routing in publish/subscribe hierarchies" on page 96.

Topic host routing in publish/subscribe clusters:

Publications from non-hosting queue managers in the cluster are routed through the hosting queue manager to any queue manager in the cluster with a matching subscription.

For an introduction to how messages are routed between queue managers in publish/subscribe hierarchies and clusters, see Distributed publish/subscribe networks.

To understand the behavior and benefits of topic host routing it is best to first understand "Direct routing in publish/subscribe clusters" on page 66.

A topic host routed publish/subscribe cluster behaves as follows:
- Clustered administered topic objects are manually defined on individual queue managers in the cluster. These are referred to as *topic host queue managers*.
- When a subscription is made on a cluster queue manager, channels are created from the subscription host queue manager to the topic host queue managers, and proxy subscriptions are created only on the queue managers that host the topic.
- When an application publishes information to a topic, the connected queue manager always forwards the publication to one queue manager that hosts the topic, which passes it on to all queue managers in the cluster that have matching subscriptions to the topic.

This process is explained in more detail in the following examples.

Topic host routing using a single topic host

For publications to flow between queue managers in a topic host routed cluster, you cluster a branch of the topic tree as described in Configuring a publish/subscribe cluster, and specify *topic host routing*.

There are a number of reasons to define a topic host routed topic object on multiple queue managers in a cluster. However, for simplicity we start with a single topic host.

The following diagram shows a queue manager cluster that is not currently used for publish/subscribe or point-to-point activities. Note that every queue manager in the cluster connects only to and from the full repository queue managers.

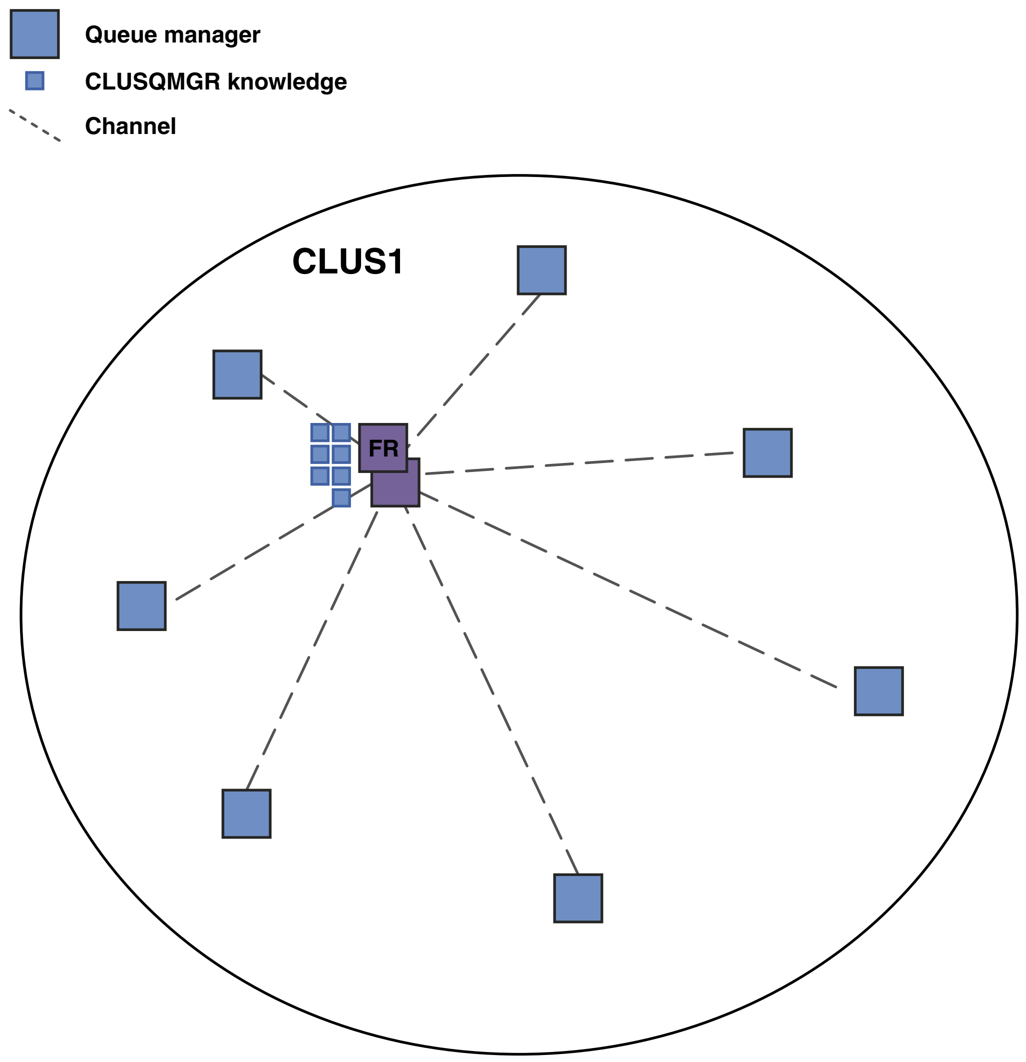


Figure 22. A queue manager cluster

In a topic host routed publish/subscribe cluster, you define the topic object on a specific queue manager in the cluster. Publish/subscribe traffic then flows through that queue manager, making it a critical queue manager in the cluster and increasing its workload. For these reasons it is not recommended to use a full repository queue manager, but to use another queue manager in the cluster. When you define the topic object on the host queue manager, knowledge of the object and its host is automatically pushed, by the full repository queue managers, to all the other queue managers in the cluster. Note that, unlike *direct routing*, each queue manager is not told about every other queue manager in the cluster.

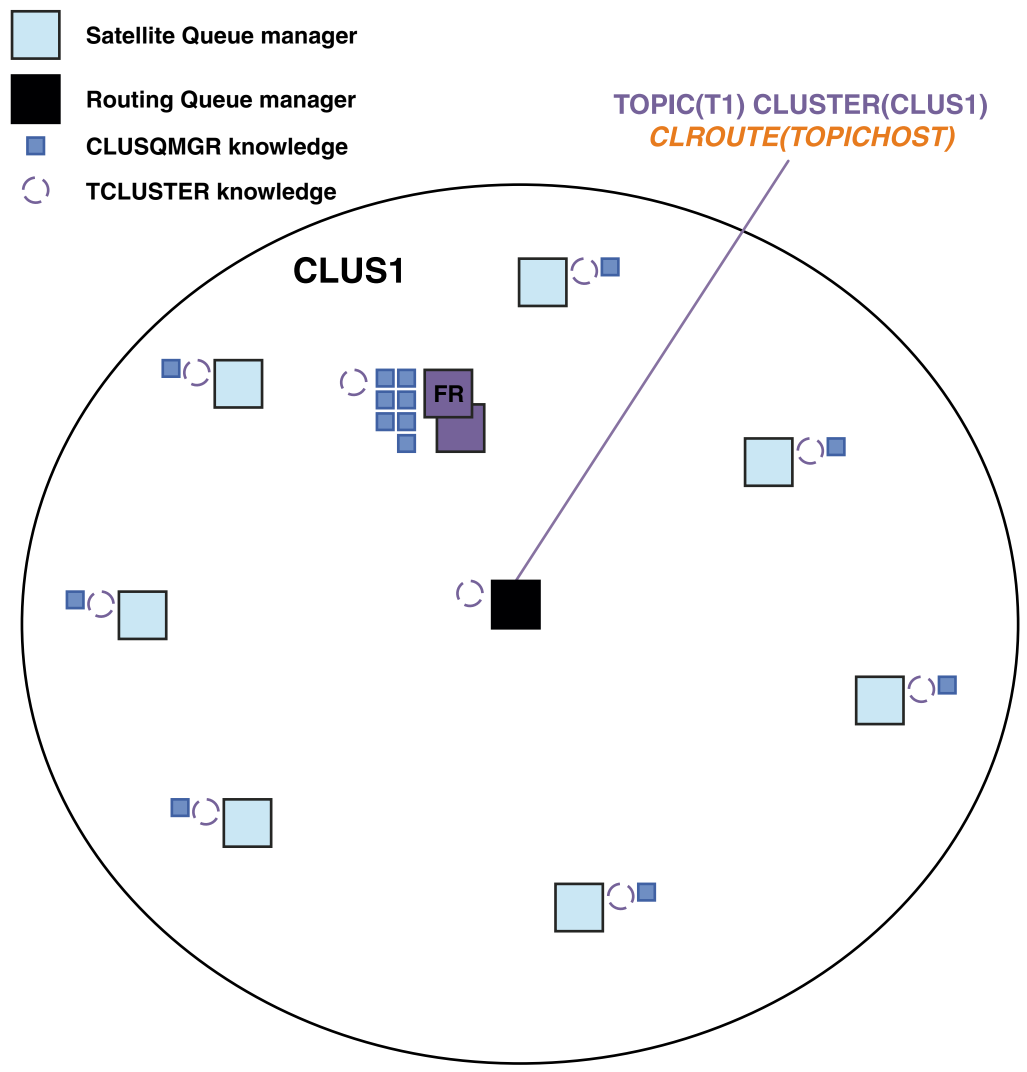


Figure 23. A topic host routed publish/subscribe cluster with one topic defined on one topic host

When a subscription is created on a queue manager, a channel is created between the subscribing queue manager and the topic host queue manager. The subscribing queue manager connects only to the topic host queue manager, and sends details of the subscription (in the form of a *proxy subscription*). The topic host queue manager does not forward this subscription information on to any further queue managers in the cluster.

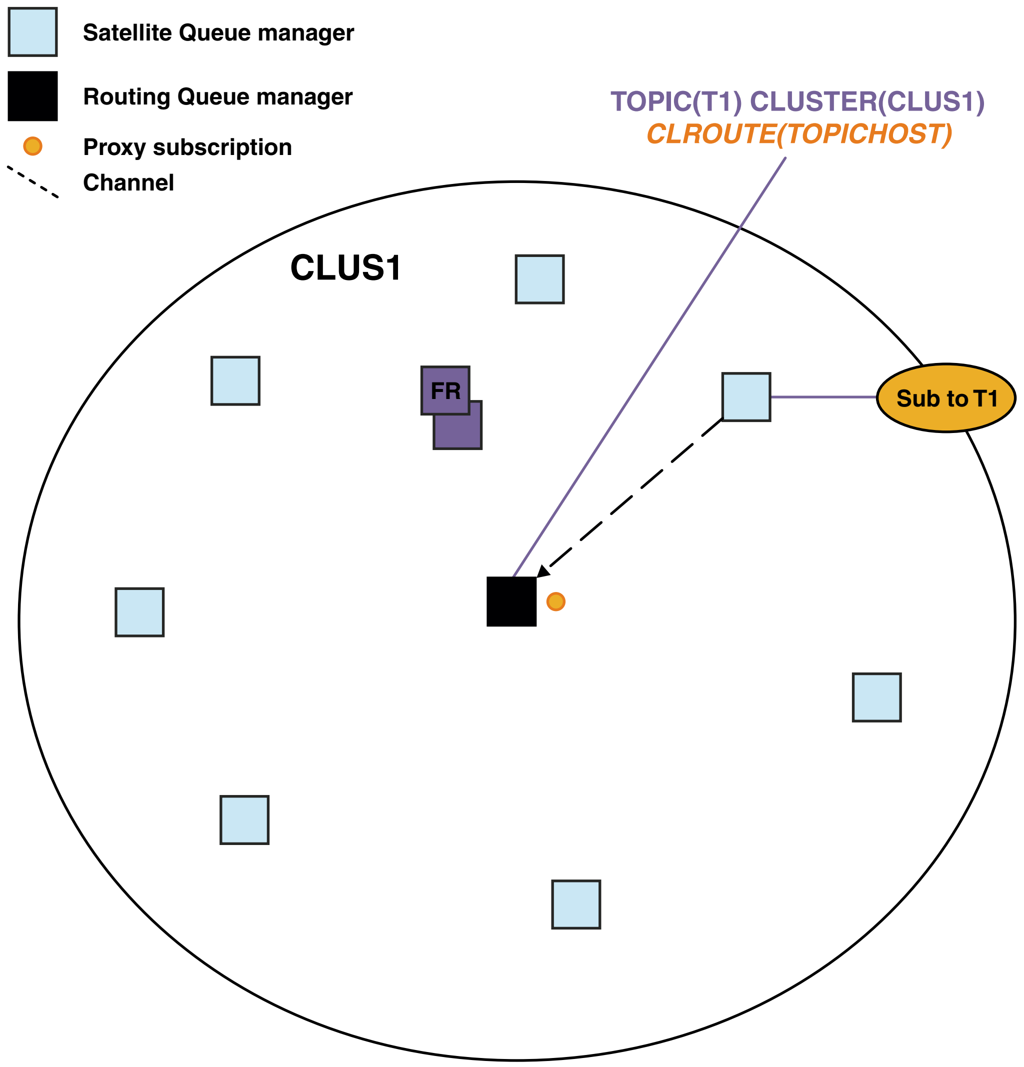


Figure 24. A topic host routed publish/subscribe cluster with one topic defined on one topic host, and one subscriber

When a publishing application connects to another queue manager and a message is published, a channel is created between the publishing queue manager and the topic host queue manager, and the message is forwarded to that queue manager. The publishing queue manager has no knowledge of any subscriptions on other queue managers in the cluster, so the message is forwarded to the topic host queue manager even if there are no subscribers to that topic in the cluster. The publishing queue manager connects only to the topic host queue manager. Publications are routed through the topic host to the subscribing queue managers, if any exist.

Subscriptions on the same queue manager as the publisher are satisfied directly, without first sending the messages to a topic host queue manager.

Note that, because of the critical role played by each topic host queue manager, you must choose queue managers that can handle the load, availability and connectivity requirements of topic hosting.

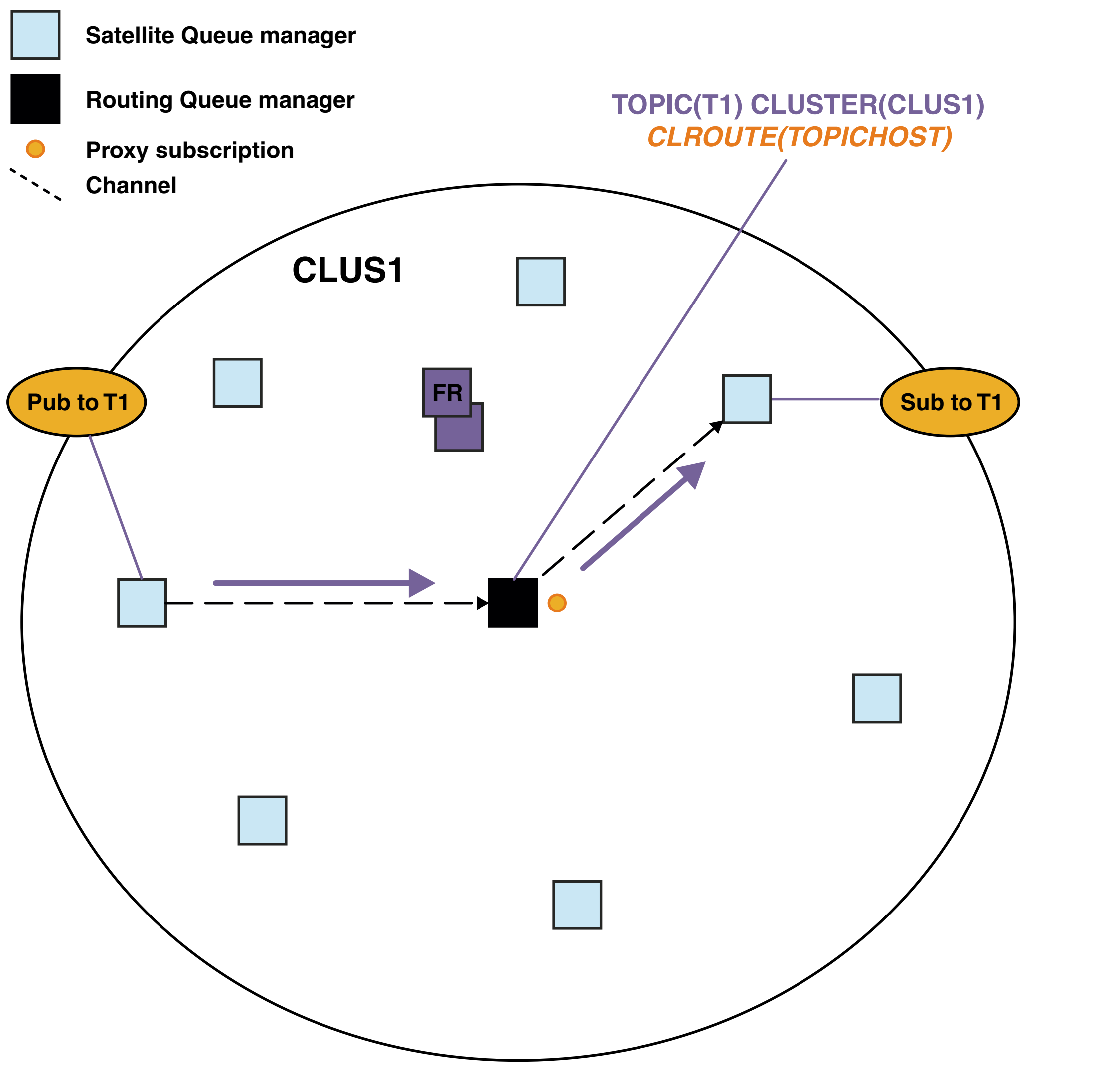


Figure 25. A topic host routed publish/subscribe cluster with one topic, one subscriber and one publisher

Dividing the topic tree across multiple queue managers

A routed topic hosting queue manager is only responsible for the subscription knowledge and publication messages that relate to the branch of the topic tree that its administered topic object is configured for. If different topics are used by different publish/subscribe applications in the cluster, you can configure different queue managers to host different clustered branches of the topic tree. This allows scaling by reducing the publication traffic, subscription knowledge and channels on each topic host queue manager in the cluster. You should use this method for distinct high volume branches of the topic tree:

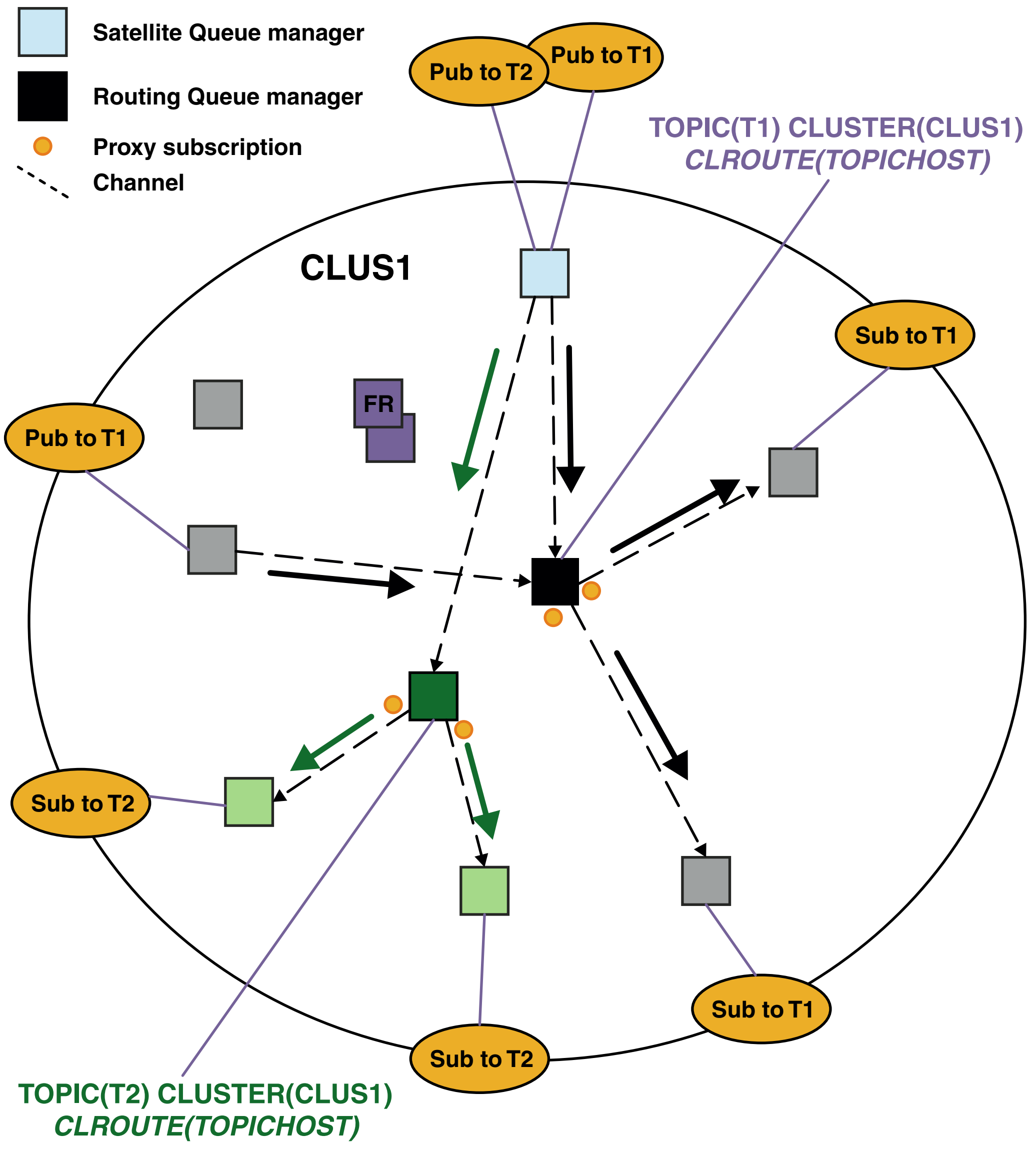


Figure 26. A topic host routed publish/subscribe cluster with two topics, each defined on one topic host

For example, using the topics described in Topic trees, if topic `T1` was configured with a topic string of `/USA/Alabama`, and topic `T2` was configured with a topic string of `/USA/Alaska`, then a message published to `/USA/Alabama/Mobile` would be routed through the queue manager hosting `T1`, and a message published to `/USA/Alaska/Juneau` would be routed through the queue manager hosting `T2`.

Note: You cannot make a single subscription span multiple clustered branches of the topic tree by using a wildcard higher in the topic tree than the points that are clustered. See Wildcard subscriptions.

Topic host routing using multiple topic hosts for a single topic

If a single queue manager has the responsibility for the routing of a topic, and that queue manager becomes unavailable or incapable of handling the workload, publications will not flow promptly to the subscriptions.

If you need greater resiliency, scalability and workload balancing than you get when you define a topic on just one queue manager, you can define a topic on more than one queue manager. Each individual message published is routed through a single topic host. When multiple matching topic host definitions exist, one of the topic hosts is chosen. The choice is made in the same way as for clustered queues. This allows messages to be routed to available topic hosts, avoiding any that are unavailable, and allows message load to be workload balanced across multiple topic host queue managers and channels. However, ordering across multiple messages is not maintained when you use multiple topic hosts for the same topic in the cluster.

The following diagram shows a topic host routed cluster in which the same topic has been defined on two queue managers. In this example, the subscribing queue managers send information about the subscribed topic to both topic host queue managers in the form of a proxy subscription:

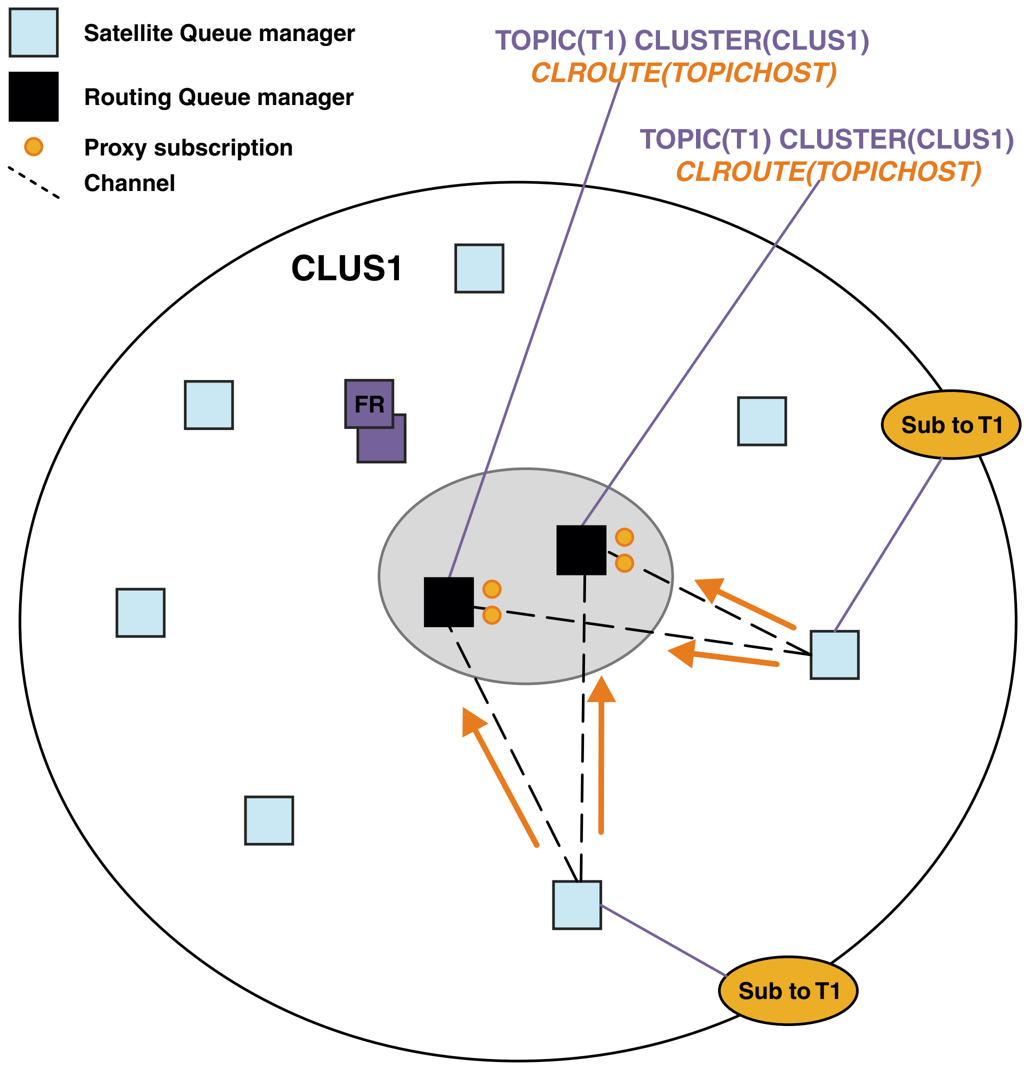


Figure 27. Creating proxy subscriptions in a multiple topic host publish/subscribe cluster

When a publication is made from a non-hosting queue manager, the queue manager sends a copy of the publication to *one* of the topic host queue managers for that topic. The system chooses the host based on the default behavior of the cluster workload management algorithm. In a typical system, this approximates to a round-robin distribution across each topic host queue manager. There is no affinity between messages from the same publishing application; this equates to using a cluster bind type of `NOTFIXED`.

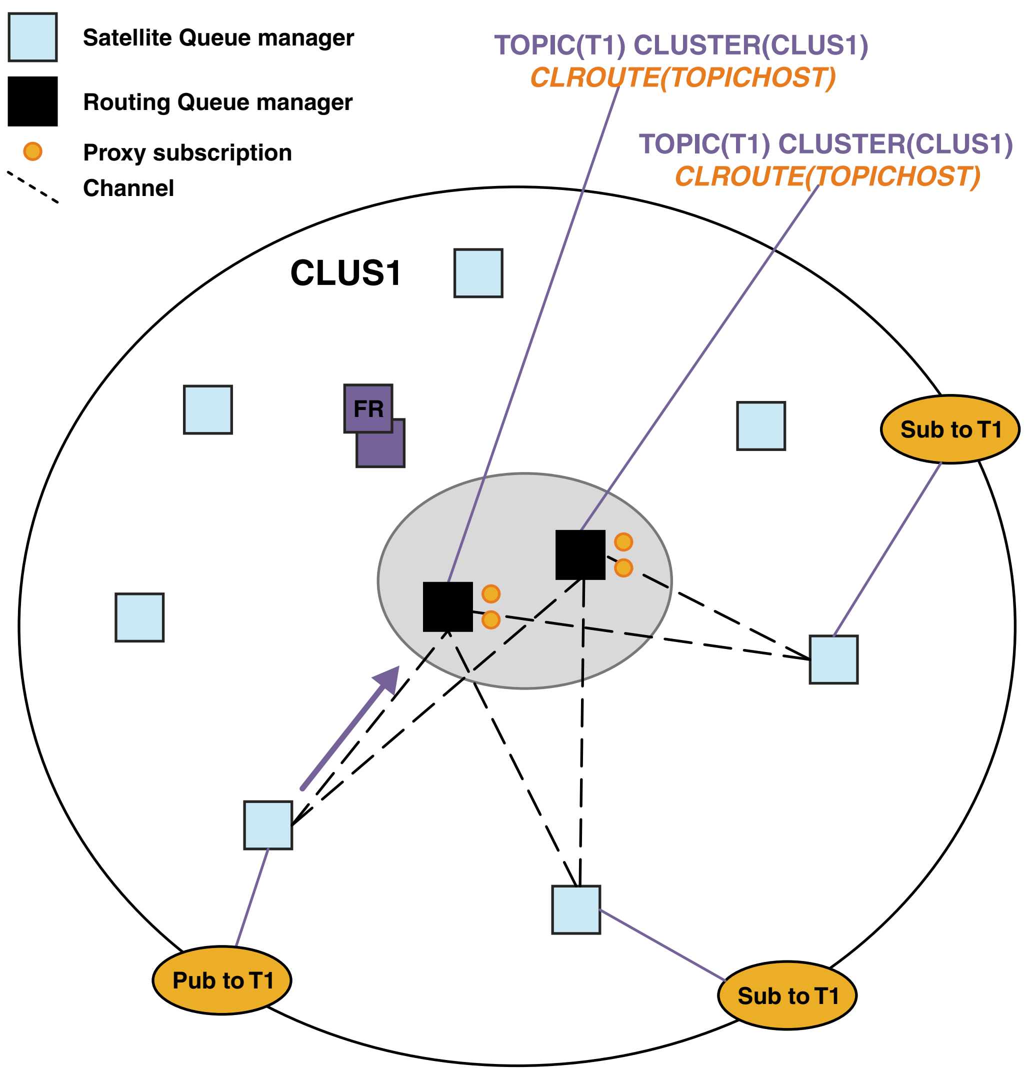


Figure 28. Receiving publications in a multiple topic host publish/subscribe cluster

Inbound publications to the chosen topic host queue manager are then forwarded to all queue managers that have registered a matching proxy subscription:

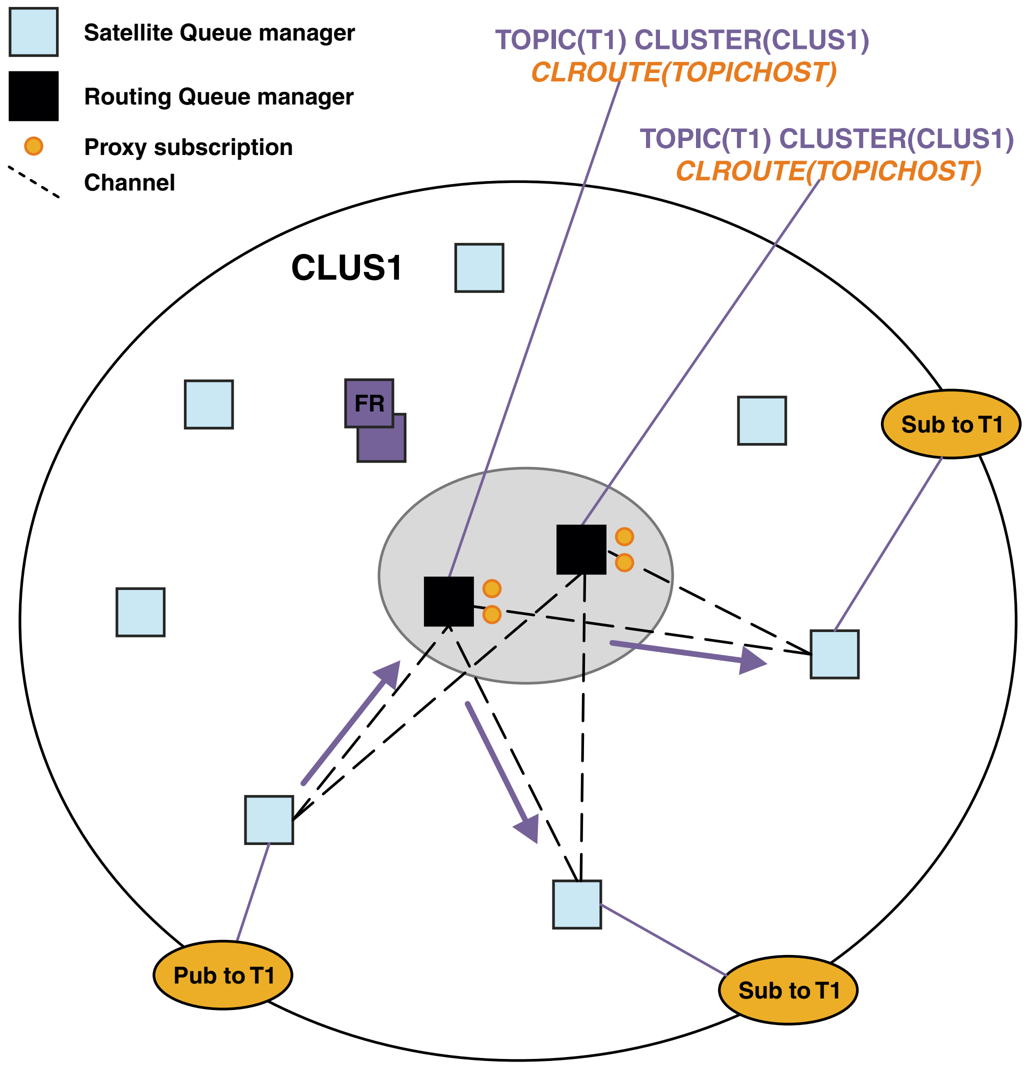


Figure 29. Routing publications to subscribers in a multiple topic host publish/subscribe cluster

Making subscriptions and publishers local to a topic host queue manager

The above examples show the routing between publishers and subscribers on queue managers that do not host administered routed topic objects. In these topologies, messages require multiple *hops* to reach the subscriptions.

Where the additional hop is not desirable, it might be appropriate to connect key publishers to topic hosting queue managers. However, if there are multiple topic hosts for a topic and only one publisher, all publication traffic will be routed through the topic host queue manager that the publisher is connected to.

Similarly, if there are key subscriptions, these could be located on a topic host queue manager. However, if there are multiple hosts of the routed topic, only a proportion of the publications will avoid the additional hop, with the remainder being routed through the other topic host queue managers first.

Topologies such as these are described further here: Topic host routing using centralized publishers or subscribers.

Note: Special planning is needed if changing the routed topic configuration when co-locating publishers or subscriptions with routed topic hosts. For example, see Adding extra topic hosts to a topic host routed cluster.

Summary and additional considerations

A topic host routed publish/subscribe cluster gives you precise control over which queue managers host each topic, and those queue managers become the *routing* queue managers for that branch of the topic tree. Moreover, queue managers without subscriptions or publishers have no need to connect with the topic host queue managers, and queue managers with subscriptions have no need to connect to queue managers that do not host a topic. This configuration can significantly reduce the number of connections between queue managers in the cluster, and the amount of information being passed between queue managers. This is especially true in large clusters where only a subset of queue managers are performing publish/subscribe work. This configuration also gives you some control over the load on individual queue managers in the cluster, so (for example) you can choose to host highly active topics on more powerful and more resilient systems. For certain configurations - notably larger clusters - it is usually a more appropriate topology than *direct routing*.

However, topic host routing also imposes certain constraints upon your system:

- System configuration and maintenance require more planning than for direct routing. You need to decide which points to cluster in the topic tree, and the location of the topic definitions in the cluster.
- Just as for direct routed topics, when a new topic host routed topic is defined, the information is pushed to the full repository queue managers, and from there direct to all members of the cluster. This event causes channels to be started to each member of the cluster from the full repositories if not already started.
- Publications are always sent to a host queue manager from a non-host queue manager, even if there are no subscriptions in the cluster. Therefore, you should use routed topics when subscriptions are typically expected to exist, or when the overhead of global connectivity and knowledge is greater than the risk of extra publication traffic.

  Note: As previously described, making publishers local to a topic host can mitigate this risk.
- Messages that are published on non-host queue managers do not go direct to the queue manager that hosts the subscription, they are always routed through a topic host queue manager. This approach can increase the total overhead to the cluster, and increase message latency and reduce performance.

  Note: As previously described, making subscriptions or publishers local to a topic host can mitigate this risk.
- Using a single topic host queue manager introduces a single point of failure for all messages that are published to a topic. You can remove this single point of failure by defining multiple topic hosts. However, having multiple hosts affects the order of published messages as received by subscriptions.
- Extra message load is incurred by topic host queue managers, because publication traffic from multiple queue managers needs to be processed by them. This load can be lessened: Either use multiple topic hosts for a single topic (in which case message ordering is not maintained), or use different queue managers to host routed topics for different branches of the topic tree.

Before you use topic host routing, explore the alternative approaches detailed in “Direct routing in publish/subscribe clusters” on page 66, and “Routing in publish/subscribe hierarchies” on page 96.

Publish/subscribe clustering: Best practices:

Using clustered topics makes extending the publish/subscribe domain between queue managers simple, but can lead to problems if the mechanics and implications are not fully understood. There are two models for information sharing and publication routing. Implement the model that best meets your individual business needs, and performs best on your chosen cluster.

The best practice information in the following sections does not provide a one size fits all solution, but rather shares common approaches to solving common problems. It assumes that you have a basic understanding of IBM MQ clusters, and of publish/subscribe messaging, and that you are familiar with the information in Distributed publish/subscribe networks and "Designing publish/subscribe clusters" on page 64.

When you use a cluster for point-to-point messaging, each queue manager in the cluster works on a need-to-know basis. That is, it only finds out about other cluster resources, such as other queue managers in the cluster and clustered queues, when applications connecting to them request to use them. When you add publish/subscribe messaging to a cluster, an increased level of sharing of information and connectivity between cluster queue managers is introduced. To be able to follow best practices for publish/subscribe clusters, you need to fully understand the implications of this change in behavior.

To allow you to build the best architecture, based on your precise needs, there are two models for information sharing and publication routing in publish/subscribe clusters: *direct routing* and *topic host routing*. To make the right choice, you need to understand both models, and the different requirements that each model satisfies. These requirements are discussed in the following sections, in conjunction with "Planning your distributed publish/subscribe network" on page 61:
- "Reasons to limit the number of cluster queue managers involved in publish/subscribe activity"
- "How to decide which topics to cluster" on page 82
- "How to size your system" on page 82
- "Publisher and subscription location" on page 83
- "Publication traffic" on page 84
- "Subscription change and dynamic topic strings" on page 84

Reasons to limit the number of cluster queue managers involved in publish/subscribe activity

There are capacity and performance considerations when you use publish/subscribe messaging in a cluster. Therefore, it is best practice to consider carefully the need for publish/subscribe activity across queue managers, and to limit it to only the number of queue managers that require it. After the minimum set of queue managers that need to publish and subscribe to topics are identified, they can be made members of a cluster that contains only them and no other queue managers.

This approach is especially useful if you have an established cluster already functioning well for point-to-point messaging. When you are turning an existing large cluster into a publish/subscribe cluster, it is a better practice to initially create a separate cluster for the publish/subscribe work where the applications can be tried, rather than using the current cluster. You can use a subset of existing queue managers that are already in one or more point-to-point clusters, and make this subset members of the new publish/subscribe cluster. However, the full repository queue managers for your new cluster must not be members of any other cluster; this isolates the additional load from the existing cluster full repositories.

If you cannot create a new cluster, and have to turn an existing large cluster into a publish/subscribe cluster, do not use a direct routed model. The topic host routed model usually performs better in larger clusters, because it generally restricts the publish/subscribe information sharing and connectivity to the set of queue managers that are actively performing publish/subscribe work, concentrating on the queue managers hosting the topics. The exception to that is if a manual refresh of the subscription information is invoked on a queue manager hosting a topic definition, at which point the topic host queue manager will connect to every queue manager in the cluster. See Resynchronization of proxy subscriptions.

If you establish that a cluster cannot be used for publish/subscribe due to its size or current load, it is good practice to prevent this cluster unexpectedly being made into a publish/subscribe cluster. Use the **PSCLUS** queue manager property to stop anyone adding a clustered topic on any queue manager in the cluster. See "Inhibiting clustered publish/subscribe" on page 91.

How to decide which topics to cluster

It is important to choose carefully which topics are added to the cluster: The higher up the topic tree these topics are, the more widespread their use becomes. This can result in more subscription information and publications being propagated than necessary. If there are multiple, distinct branches of the topic tree, where some need to be clustered and some do not, create administered topic objects at the root of each branch that needs clustering and add those to the cluster. For example, if branches `/A`, `/B` and `/C` need clustering, define a separate clustered topic objects for each branch.

Note: The system prevents you from nesting clustered topic definitions in the topic tree. You are only permitted to cluster topics at one point in the topic tree for each sub branch. For example, you cannot define clustered topic objects for `/A` and for `/A/B`. Nesting clustered topics can lead to confusion over which clustered object applies to which subscription, especially when subscriptions are using wildcards. This is even more important when using topic host routing, where routing decisions are precisely defined by your allocation of topic hosts.

If clustered topics must be added high up the topic tree, but some branches of the tree below the clustered point do not require the clustered behavior, you can use the subscription and publication scope attributes to reduce the level of subscription and publication sharing for further topics.

You should not put the topic root node into the cluster without considering the behavior that is seen. Make global topics obvious where possible, for example by using a high-level qualifier in the topic string: `/global` or `/cluster`.

There is a further reason for not wanting to make the root topic node clustered. This is because every queue manager has a local definition for the root node, the SYSTEM.BASE.TOPIC topic object. When this object is clustered on one queue manager in the cluster, all other queue managers are made aware of it. However, when a local definition of the same object exists, its properties override the cluster object. This results in those queue managers acting as if the topic was not clustered. To resolve this, you would need to cluster every definition of SYSTEM.BASE.TOPIC. You could do this for direct routed definitions, but not for topic host routed definitions, because it causes every queue manager to become a topic host.

How to size your system

Publish/subscribe clusters typically result in a different pattern of cluster channels to point-to-point messaging in a cluster. The point-to-point model is an 'opt in' one, but publish/subscribe clusters have a more indiscriminate nature with subscription fan-out, especially when using direct routed topics. Therefore, it is important to identify which queue managers in a publish/subscribe cluster will use cluster channels to connect to other queue managers, and under what circumstances.

The following table lists the typical set of cluster sender and receiver channels expected for each queue manager in a publish/subscribe cluster under normal running, dependent on the queue manager role in the publish/subscribe cluster.

Table 5. Cluster sender and receiver channels for each routing method.

Queue manager role	Direct cluster receivers	Direct cluster senders	Topic cluster receivers	Topic cluster senders
Full repository	AllQmgrs	AllQmgrs	AllQmgrs	AllQMgrs
Host of topic definition	n/a	n/a	AllSubs+AllPubs (1)	AllSubs (1)
Subscriptions created	AllPubs (1)	AllQMgrs	AllHosts	AllHosts
Publishers connected	AllSubs (1)	AllSubs (1)	AllHosts	AllHosts
No publishers or subscribers	AllSubs (1)	None (1)	None (2)	None (2)

Key:

AllQmgrs
A channel to and from every queue manager in the cluster.

AllSubs
A channel to and from every queue manager where a subscription has been created.

AllPubs
A channel to and from every queue manager where a publishing application has been connected.

AllHosts
A channel to and from every queue manager where a definition of the clustered topic object has been configured.

None No channels to or from other queue managers in the cluster for the sole purpose of publish/subscribe messaging.

Notes:

1. If a queue manager refresh of proxy subscriptions is made from this queue manager, a channel to and from all other queue managers in the cluster might be automatically created.
2. If a queue manager refresh of proxy subscriptions is made from this queue manager, a channel to and from any other queue managers in the cluster that host a definition of a clustered topic might be automatically created.

The previous table shows that topic host routing typically uses significantly less cluster sender and receiver channels than direct routing. If channel connectivity is a concern for certain queue managers in a cluster, for reasons of capacity or ability to establish certain channels (for example, through firewalls), topic host routing is therefore a preferred solution.

Publisher and subscription location

Clustered publish/subscribe enables messages published on one queue manager to be delivered to subscriptions on any other queue manager in the cluster. As for point-to-point messaging, the cost of transmitting messages between queue managers can be detrimental to performance. Therefore you should consider creating subscriptions to topics on the same queue managers as where messages are being published.

When using topic host routing within a cluster, it is important to also consider the location of the subscriptions and publishers with respect to the topic hosting queue managers. When the publisher is not connected to a queue manager that is a host of the clustered topic, messages published are always sent to a topic hosting queue manager. Similarly, when a subscription is created on a queue manager that is not a topic host for a clustered topic, messages published from other queue managers in the cluster are always sent to a topic hosting queue manager first. More specifically, if the subscription is located on a queue manager that hosts the topic, but there is one or more other queue managers that also host that same topic, a proportion of publications from other queue managers are routed through those other topic hosting queue managers. See Topic host routing using centralized publishers or subscribers for more information on designing a topic host routed publish/subscribe cluster to minimize the distance between publishers and subscriptions.

Publication traffic

Messages published by an application connected to one queue manager in a cluster are transmitted to subscriptions on other queue managers using cluster sender channels.

When you use direct routing, the messages published take the shortest path between queue managers. That is, they go direct from the publishing queue manager to each of the queue managers with subscriptions. Messages are not transmitted to queue managers that do not have subscriptions for the topic. See Proxy subscriptions in a publish/subscribe network.

Where the rate of publication messages between any one queue manager and another in the cluster is high, the cluster channel infrastructure between those two points must be able to maintain the rate. This might involve tuning the channels and transmission queue being used.

When you use topic host routing, each message published on a queue manager that is not a topic host is transmitted to a topic host queue manager. This is independent of whether one or more subscriptions exist anywhere else in the cluster. This introduces further factors to consider in planning:
- Is the additional latency of first sending each publication to a topic host queue manager acceptable?
- Can each topic host queue manager sustain the inbound and outbound publication rate? Consider a system with publishers on many different queue managers. If they all send their messages to a very small set of topic hosting queue managers, those topic hosts might become a bottleneck in processing those messages and routing them on to subscribing queue managers.
- Is it expected that a significant proportion of the published messages will not have a matching subscriber? If so, and the rate of publishing such messages is high, it might be best to make the publisher's queue manager a topic host. In that situation, any published message where no subscriptions exist in the cluster will not be transmitted to any other queue managers.

These problems might also be eased by introducing multiple topic hosts, to spread the publication load across them:
- Where there are multiple distinct topics, each with a proportion of the publication traffic, consider hosting them on different queue managers.
- If the topics cannot be separated onto different topic hosts, consider defining the same topic object on multiple queue managers. This results in publications being workload balanced across each of them for routing. However, this is only appropriate when publication message ordering is not required.

Subscription change and dynamic topic strings

Another consideration is the effect on performance of the system for propagating proxy subscriptions. Typically, a queue manager sends a proxy subscription message to certain other queue managers in the cluster when the first subscription for a specific clustered topic string (not just a configured topic object) is created on that queue manager. Similarly, a proxy subscription deletion message is sent when the last subscription for a specific clustered topic string is deleted.

For direct routing, each queue manager with subscriptions sends those proxy subscriptions to every other queue manager in the cluster. For topic host routing, each queue manager with subscriptions only sends the proxy subscriptions to each queue manager that hosts a definition for that clustered topic. Therefore, with direct routing, the more queue managers there are in the cluster, the higher the overhead of maintaining proxy subscriptions across them. Whereas, with topic host routing, the number of queue managers in the cluster is not a factor.

In both routing models, if a publish/subscribe solution consists of many unique topic strings being subscribed to, or the topics on a queue manager in the cluster are frequently being subscribed and unsubscribed, a significant overhead will be seen on that queue manager, caused by constantly generating messages distributing and deleting the proxy subscriptions. With direct routing, this is compounded by the need to send these messages to every queue manager in the cluster.

If the rate of change of subscriptions is too high to accommodate, even within a topic host routed system, see Subscription performance in publish/subscribe networks for information about ways to reduce proxy subscription overhead.

Defining cluster topics:

Cluster topics are administrative topics with the **`cluster`** attribute defined. Information about cluster topics is pushed to all members of a cluster, and combined with local topics to create portions of a topic space that spans multiple queue managers. This enables messages published on a topic on one queue manager to be delivered to subscriptions of other queue managers in the cluster.

When you define a cluster topic on a queue manager, the cluster topic definition is sent to the full repository queue managers. The full repositories then propagate the cluster topic definition to all queue managers within the cluster, making the same cluster topic available to publishers and subscribers at any queue manager in the cluster. The queue manager on which you create a cluster topic is known as a cluster topic host. The cluster topic can be used by any queue manager in the cluster, but any modifications to a cluster topic must be made on the queue manager where that topic is defined (the host) at which point the modification is propagated to all members of the cluster through the full repositories.

When you use direct routing, the location of the clustered topic definition does not directly affect the behavior of the system, because all queue managers in the cluster use the topic definition in the same way. You should therefore define the topic on any queue manager that will be a member of the cluster for as long as the topic is needed, and that is on a system reliable enough to regularly be in contact with the full repository queue managers.

When you use topic host routing, the location of the clustered topic definition is very important, because other queue managers in the cluster create channels to this queue manager and send subscription information and publications to it. To choose the best queue manager to host the topic definition, you need to understand topic host routing. See "Topic host routing in publish/subscribe clusters" on page 71.

If you have a clustered topic, and a local topic object, then the local topic takes precedence. See "Multiple cluster topic definitions of the same name" on page 88.

For information about the commands to use to display cluster topics, see the related information.

Clustered topic inheritance

Typically, publishing and subscribing applications in a clustered publish/subscribe topology expect to work the same, no matter which queue manager in the cluster they are connected to. This is why clustered administered topic objects are propagated to every queue manager in the cluster.

An administered topic object inherits its behavior from other administered topic objects higher in the topic tree. This inheritance occurs when an explicit value has not been set for a topic parameter.

In the case of clustered publish/subscribe, it is important to consider such inheritance because it introduces the possibility that publishers and subscribers will behave differently depending on which queue manager they connect to. If a clustered topic object leaves any parameters to inherit from higher topic objects, the topic might behave differently on different queue managers in the cluster. Similarly, locally defined topic objects defined below a clustered topic object in the topic tree will mean those lower topics are still clustered, but the local object might change its behavior in some way that differs from other queue managers in the cluster.

Wildcard subscriptions

Proxy subscriptions are created when local subscriptions are made to a topic string that resolves at, or below, a clustered topic object. If a wildcard subscription is made higher in the topic hierarchy than any cluster topic, it does not have proxy subscriptions sent around the cluster for the matching cluster topic, and therefore receives no publications from other members of the cluster. It does however receive publications from the local queue manager.

However, if another application subscribes to a topic string that resolves to or below the cluster topic, proxy subscriptions are generated and publications are propagated to this queue manager. On arrival the original, higher wildcard subscription is considered a legitimate recipient of those publications and receives a copy. If this behavior is not required, set **`WILDCARD(BLOCK)`** on the clustered topic. This makes the original wildcard not be considered a legitimate subscription, and stops it receiving any publications (local, or from elsewhere in the cluster) on the cluster topic, or its subtopics.

Related information:

Working with administrative topics

Working with subscriptions

DISPLAY TOPIC

DISPLAY TPSTATUS

DISPLAY SUB

Cluster topic attributes:

When a topic object has the cluster name attribute set, the topic definition is propagated across all queue managers in the cluster. Each queue manager uses the propagated topic attributes to control the behavior of publish/subscribe applications.

A topic object has a number of attributes that apply to publish/subscribe clusters. Some control the general behavior of the publishing and subscribing applications and some control how the topic is used across the cluster.

A clustered topic object definition must be configured in a way that all queue managers in the cluster can correctly use it.

For example if the model queues to be used for managed subscriptions (`MDURMDL` and `MNDURMDL`) are set to a non-default queue name, that named model queue must be defined on all queue managers where managed subscriptions will be created.

Similarly, if any attribute is set to `ASPARENT`, the behavior of the topic will be dependent on the higher nodes in the topic tree (see Administrative topic objects) on each individual queue manager in the cluster. This might result in different behavior when publishing or subscribing from different queue managers.

The main attributes that directly relate to publish/subscribe behavior across the cluster are as follows:

CLROUTE

This parameter controls the routing of messages between queue managers where publishers are connected, and queue managers where matching subscriptions exist.

- You configure the route to be either direct between these queue managers, or through a queue manager that hosts a definition of the clustered topic. See Publish/subscribe clusters for more details.

- You cannot change the **CLROUTE** while the **CLUSTER** parameter is set. To change the **CLROUTE**, first set the **CLUSTER** property to be blank. This stops applications that use the topic from behaving in a clustered manner. This in turn results in a break in publications being delivered to subscriptions, so you should also quiesce publish/subscribe messaging while making the change.

PROXYSUB

This parameter controls when proxy subscriptions are made.

- FIRSTUSE is the default value, and causes proxy subscriptions to be sent in response to local subscriptions on a queue manager in a distributed publish/subscribe topology, and canceled when no longer required. For details about why you might want to change this attribute from the default value of FIRSTUSE, see Individual proxy subscription forwarding and *publish everywhere* .
- To enable *publish everywhere*, you set the **PROXYSUB** parameter to FORCE for a high-level topic object. This results in a single wildcard proxy subscription that matches all topics below this topic object in the topic tree.

Note: Setting the **PROXYSUB(FORCE)** attribute in a large or busy publish/subscribe cluster can result in excessive load on system resources. The **PROXYSUB(FORCE)** attribute is propagated to every queue manager, not just the queue manager that the topic was defined on. This causes every queue manager in the cluster to create a wildcarded proxy subscription.

A copy of a message to this topic, published on any queue manager in the cluster, is sent to every queue manager in the cluster - either directly, or through a topic host queue manager, depending on the **CLROUTE** setting.

When the topic is direct routed, every queue manager creates cluster sender channels to every other queue manager. When the topic is topic host routed, channels to each topic host queue manager are created from every queue manager in the cluster.

For more information about the **PROXYSUB** parameter when used in clusters, see Direct routed publish/subscribe performance.

PUBSCOBE and SUBSCOPE

These parameters determine whether this queue manager propagates publications to queue managers in the topology (publish/subscribe cluster or hierarchy) or restricts the scope to just its local queue manager. You can do the equivalent job programmatically using MQPMO_SCOPE_QMGR and MQSO_SCOPE_QMGR.

PUBSCOPE

If a cluster topic object is defined with **PUBSCOPE(QMGR)**, the definition is shared with the cluster, but the scope of publications that are based on that topic is local only and they are not sent to other queue managers in the cluster.

SUBSCOPE

If a cluster topic object is defined with **SUBSCOPE(QMGR)**, the definition is shared with the cluster, but the scope of subscriptions that are based on that topic is local only, therefore no proxy subscriptions are sent to other queue managers in the cluster.

These two attributes are commonly used together to isolate a queue manager from interacting with other members of the cluster on particular topics. The queue manager neither publishes or receives publications on those topics to and from other members of the cluster. This situation does not prevent publication or subscription if topic objects are defined on subtopics.

Setting **SUBSCOPE** to QMGR on a local definition of a topic does not prevent other queue managers in the cluster from propagating their proxy subscriptions to the queue manager if they are using a clustered version of the topic, with **SUBSCOPE(ALL)**. However, if the local definition also sets **PUBSCOPE** to QMGR those proxy subscriptions are not sent publications from this queue manager.

Related information:
Publication scope
Subscription scope

Multiple cluster topic definitions of the same name:

You can define the same named cluster topic object on more than one queue manager in the cluster, and in certain scenarios this enables specific behavior. When multiple cluster topic definitions exist of the same name, the majority of properties should match. If they do not, errors or warnings are reported depending on the significance of the mismatch.

In general, if there is a mismatch in the properties of multiple cluster topic definitions, warnings are issued and one of the topic object definitions is used by each queue manager in the cluster. Which definition is used by each queue manager is not deterministic, or consistent across the queue managers in the cluster. Such mismatches should be resolved as quickly as possible.

During cluster setup or maintenance you sometimes need to create multiple cluster topic definitions that are not identical. However this is only ever useful as a temporary measure, and it is therefore treated as a potential error condition.

When mismatches are detected, the following warning messages are written to each queue manager's error log:

- Multi On Multiplatforms, AMQ9465 and AMQ9466.
- z/OS On z/OS, CSQX465I and CSQX466I.

The chosen properties for any topic string on each queue manager can be determined by viewing topic status rather than the topic object definitions, for example by using **DISPLAY TPSTATUS**.

In some situations, a conflict in configuration properties is severe enough to stop the topic object being created, or to cause the mismatched objects to be marked as invalid and not propagated across the cluster (See **CLSTATE** in DISPLAY TOPIC). These situations occur when there is a conflict in the cluster routing property (**CLROUTE**) of the topic definitions. Additionally, due to the importance of consistency across topic host routed definitions, further inconsistencies are rejected as detailed in subsequent sections of this article.

If the conflict is detected at the time that the object is defined, the configuration change is rejected. If detected later by the full repository queue managers, the following warning messages are written to the queue managers error logs:

- Multi On Multiplatforms: AMQ9879
- z/OS On z/OS: CSQX879E.

When multiple definitions of the same topic object are defined in the cluster, a locally defined definition takes precedence over any remotely defined one. Therefore, if any differences exist in the definitions, the queue managers hosting the multiple definitions behave differently from each other.

The effect of defining a non-cluster topic with the same name as a cluster topic from another queue manager

It is possible to define an administered topic object that is not clustered on a queue manager that is in a cluster, and simultaneously define the same named topic object as a clustered topic definition on a different queue manager. In this case, the locally defined topic object takes precedence over all remote definitions of the same name.

This has the effect of preventing the clustering behavior of the topic when used from this queue manager. That is, subscriptions might not receive publications from remote publishers, and messages from publishers might not be propagated to remote subscriptions in the cluster.

Careful consideration should be given before configuring such a system, because this can lead to confusing behavior.

Note: If an individual queue manager needs to prevent publications and subscriptions from propagating around the cluster, even when the topic has been clustered elsewhere, an alternative approach is to set the publication and subscription scopes to only the local queue manager. See "Cluster topic attributes" on page 86.

Multiple cluster topic definitions in a direct routed cluster

For direct routing, you do not usually define the same cluster topic on more than one cluster queue manager. This is because direct routing makes the topic available at all queue managers in the cluster, no matter which queue manager it was defined on. Moreover, adding multiple cluster topic definitions significantly increases system activity and administrative complexity, and with increased complexity comes a greater chance of human error:
- Each definition results in an additional cluster topic object being pushed out to the other queue managers in the cluster, including the other cluster topic host queue managers.
- All definitions for a specific topic in a cluster must be identical, otherwise it is difficult to work out which topic definition is being used by a queue manager.

It is also not essential that the sole host queue manager is continually available for the topic to function correctly across the cluster, because the cluster topic definition is cached by the full repository queue managers and by all other queue managers in their partial cluster repositories. For more information, see Availability of topic host queue managers that use direct routing.

For a situation in which you might need to temporarily define a cluster topic on a second queue manager, for example when the existing host of the topic is to be removed from the cluster, see Moving a cluster topic definition to a different queue manager.

If you need to alter a cluster topic definition, take care to modify it at the same queue manager it was defined on. Attempting to modify it from another queue manager might accidentally create a second definition of the topic with conflicting topic attributes.

Multiple cluster topic definitions in a topic host routed cluster

When a cluster topic is defined with a cluster route of *topic host*, the topic is propagated across all queue managers in the cluster just as for *direct* routed topics. Additionally, all publish/subscribe messaging for that topic is routed through the queue managers where that topic is defined. Therefore the location and number of definitions of the topic in the cluster becomes important (see "Topic host routing in publish/subscribe clusters" on page 71).

To ensure adequate availability and scalability it is useful, if possible, to have multiple topic definitions. See Availability of topic host queue managers that use topic host routing.

When adding or removing additional definitions of a *topic host* routed topic in a cluster, you should consider the flow of messages at the time of the configuration change. If messages are being published in the cluster to the topic at the time of the change, a staged process is required to add or remove a topic definition. See Moving a cluster topic definition to a different queue manager and Adding extra topic hosts to a topic host routed cluster.

As previously explained, the properties of the multiple definitions should match, with the possible exception of the **PUB** parameter, as described in the next section. When publications are routed through topic host queue managers it is even more important for multiple definitions to be consistent. Therefore, an inconsistency detected in either the topic string or cluster name is rejected if one or more of the topic definitions has been configured for topic host cluster routing.

Note: Cluster topic definitions are also rejected if an attempt is made to configure them above or below another topic in the topic tree, where the existing clustered topic definition is configured for topic host routing. This prevents ambiguity in the routing of publications with respect to wildcarded subscriptions.

Special handling for the PUB parameter

The **PUB** parameter is used to control when applications can publish to a topic. In the case of topic host routing in a cluster, it can also control which topic host queue managers are used to route publications. For this reason it is permitted to have multiple definitions of the same topic object in the cluster, with different settings for the PUB parameter.

If multiple remote clustered definitions of a topic have different settings for this parameter, the topic allows publications to be sent and delivered to subscriptions if the following conditions are met:
- There is not a matching topic object defined on the queue manager that the publisher is connected to that is set to `PUB(DISABLED)`.
- One or more of the multiple topic definitions in the cluster is set to `PUB(ENABLED)`, or one or more of the multiple topic definitions is set to `PUB(ASPARENT)` and the local queue managers where the publisher is connected and the subscription defined are set to `PUB(ENABLED)` at a higher point in the topic tree.

For topic host routing, when messages are published by applications connected to queue managers that are not topic hosts, messages are only routed to the topic hosting queue managers where the **PUB** parameter has not explicitly been set to `DISABLED`. You can therefore use the `PUB(DISABLED)` setting to quiesce message traffic through certain topic hosts. You might want to do this to prepare for maintenance or removal of a queue manager, or for the reasons described in Adding extra topic hosts to a topic host routed cluster.

Availability of cluster topic host queue managers:

Design your publish/subscribe cluster to minimize the risk that, should a topic host queue manager become unavailable, the cluster will no longer be able to process traffic for the topic. The effect of a topic host queue manager becoming unavailable depends on whether the cluster is using topic host routing or direct routing.

Availability of topic host queue managers that use direct routing

For direct routing, you do not usually define the same cluster topic on more than one cluster queue manager. This is because direct routing makes the topic available at all queue managers in the cluster, no matter which queue manager it was defined on.See Multiple cluster topic definitions in a direct routed cluster.

In a cluster, whenever the host of a clustered object (for example a clustered queue or clustered topic) becomes unavailable for a prolonged period of time, the other members of the cluster will eventually expire the knowledge of those objects. In the case of a clustered topic, if the cluster topic host queue manager becomes unavailable, the other queue managers continue to process publish/subscribe requests for the topic in a direct clustered way (that is, sending publications to subscriptions on remote queue managers) for at least 60 days from when the topic hosting queue manager was last in communication with the full repository queue managers. If the queue manager on which you defined the cluster topic object is never made available again, then eventually the cached topic objects on the other queue managers are deleted and the topic reverts to a local topic, in which case subscriptions cease to receive publications from applications connected to remote queue managers.

With the 60 day period to recover the queue manager on which you define a cluster topic object, there is little need to take special measures to guarantee that a cluster topic host remains available (note, however, that any subscriptions defined on the unavailable cluster topic host do not remain available). The 60 day period is sufficient to cater for technical problems, and is likely to be exceeded only because of administrative errors. To mitigate that possibility, if the cluster topic host is unavailable, all members of the cluster write error log messages hourly, stating that their cached cluster topic object was not refreshed. Respond to these messages by making sure that the queue manager on which the cluster topic object is defined, is running. If it is not possible to make the cluster topic host queue manager available again, define the same clustered topic definition, with exactly the same attributes, on another queue manager in the cluster.

Availability of topic host queue managers that use topic host routing

For topic host routing, all publish/subscribe messaging for a topic is routed through the queue managers where that topic is defined. For this reason, it is very important to consider the continual availability of these queue managers in the cluster. If a topic host becomes unavailable, and no other host exists for the topic, traffic from publishers to subscribers on different queue managers in the cluster immediately halts for the topic. If additional topic hosts are available, the cluster queue managers route new publication traffic through these topic hosts, providing continuous availability of message routes.

As for direct topics, after 60 days, if the first topic host is still unavailable, knowledge of that topic host's topic is removed from the cluster. If this is the last remaining definition for this topic in the cluster, all other queue managers cease to forward publications to any topic host for routing.

To ensure adequate availability and scalability it is therefore useful, if possible, to define each topic on at least two cluster queue managers. This gives protection against any given topic host queue manager becoming unavailable. See also Multiple cluster topic definitions in a topic host routed cluster.

If you cannot configure multiple topic hosts (for example because you need to preserve message ordering), and you cannot configure just one topic host (because the availability of a single queue manager must not affect the flow of publications to subscriptions across all queue managers in the cluster), consider configuring the topic as a direct routed topic. This avoids reliance on a single queue manager for the whole cluster, but does still require each individual queue manager to be available in order for it to process locally hosted subscriptions and publishers.

Inhibiting clustered publish/subscribe:

Introducing the first direct routed clustered topic into a cluster forces every queue manager in the cluster to become aware of every other queue manager, and potentially causes them to create channels to each other. If this is not desirable, you should instead configure topic host routed publish/subscribe. If the existence of a direct routed clustered topic might jeopardize the stability of the cluster, due to scaling concerns of each queue manager, you can completely disable clustered publish/subscribe functionality by setting **PSCLUS** to `DISABLED` on every queue manager in the cluster.

As described in"Direct routing in publish/subscribe clusters" on page 66, when you introduce a direct routed clustered topic into a cluster, all partial repositories are automatically notified of all other members of the cluster. The clustered topic might also create subscriptions at all other nodes (for example, where **PROXYSUB(FORCE)** is specified) and cause large numbers of channels to be started from a queue manager, even when there are no local subscriptions. This puts an immediate additional load on each queue manager in the cluster. For a cluster that contains many queue managers, this can cause a significant reduction in performance. Therefore the introduction of direct routed publish/subscribe to a cluster must be carefully planned.

When you know that a cluster cannot accommodate the overheads of direct routed publish/subscribe, you can instead use topic host routed publish/subscribe. For an overview of the differences, see "Designing publish/subscribe clusters" on page 64.

If you prefer to completely disable publish/subscribe functionality for the cluster, you can do so by setting the queue manager attribute **PSCLUS** to `DISABLED` on every queue manager in the cluster. This setting disables both direct routed and topic host routed publish/subscribe in the cluster, by modifying three aspects of queue manager functionality:

- An administrator of this queue manager is no longer able to define a Topic object as clustered.
- Incoming topic definitions or proxy subscriptions from other queue managers are rejected, and a warning message is logged to inform the administrator of incorrect configuration.
- Full repositories no longer automatically share information about every queue manager with all other partial repositories when they receive a topic definition.

Although **PSCLUS** is a parameter of each individual queue manager in a cluster, it is not intended to selectively disable publish/subscribe in a subset of queue managers in the cluster. If you selectively disable in this way, you will see frequent error messages. This is because proxy subscriptions and topic definitions are constantly seen and rejected if a topic is clustered on a queue manager where **PSCLUS** is enabled.

You should therefore aim to set **PSCLUS** to `DISABLED` on every queue manager in the cluster. However, in practice this state can be difficult to achieve and maintain, for example queue managers can join and leave the cluster at any time. At the very least, you must ensure that **PSCLUS** is set to `DISABLED` on all full repository queue managers. If you do this, and a clustered topic is subsequently defined on an `ENABLED` queue manager in the cluster, this does not cause the full repositories to inform every queue manager of every other queue manager, and so your cluster is protected from potential scaling issues across all queue managers. In this scenario, the origin of the clustered topic is reported in the error logs of the full repository queue managers.

If a queue manager participates in one or more publish/subscribe clusters, and also one or more point-to-point clusters, you must set **PSCLUS** to `ENABLED` on that queue manager. For this reason, when overlapping a point-to-point cluster with a publish subscribe cluster, you should use a separate set of full repositories in each cluster. This approach allows topic definitions and information about every queue manager to flow only in the publish/subscribe cluster.

To avoid inconsistent configurations when you change **PSCLUS** from `ENABLED` to `DISABLED`, no clustered topic objects can exist in any cluster of which this queue manager is a member. Any such topics, even remotely defined ones, must be deleted before changing **PSCLUS** to `DISABLED`.

For more information about **PSCLUS**, see ALTER QMGR (PSCLUS).

Related information:
Direct routed publish/subscribe cluster performance

Publish/subscribe and multiple clusters:

A single queue manager can be a member of more than one cluster. This arrangement is sometimes known as *overlapping clusters*. Through such an overlap, clustered queues can be made accessible from multiple clusters, and point-to-point message traffic can be routed from queue managers in one cluster to queue managers in another cluster. Clustered topics in publish/subscribe clusters do not provide the same capability. Therefore, their behavior must be clearly understood when using multiple clusters.

Unlike for a queue, you cannot associate a topic definition with more than one cluster. The scope of a clustered topic is confined to those queue managers in the same cluster as the topic is defined for. This allows publications to be propagated to subscriptions only on those queue managers in the same cluster.

A queue manager's topic tree

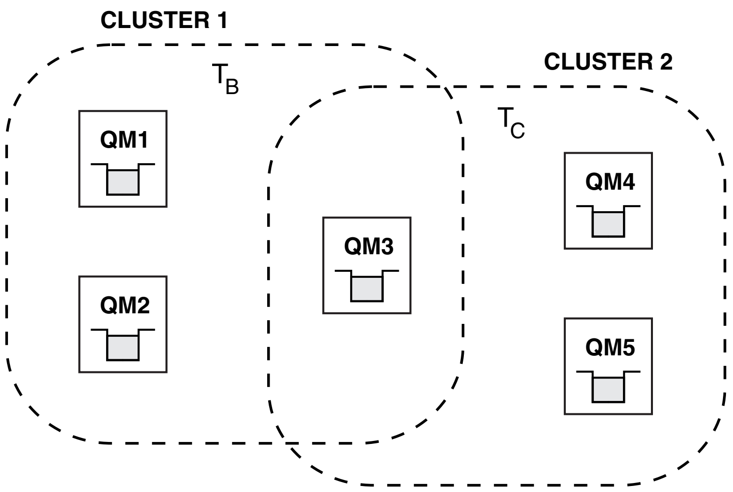


Figure 30. Overlapping clusters: Two clusters each subscribing to different topics

When a queue manager is a member of multiple clusters it is made aware of all clustered topics defined in each of those clusters. For example, in the previous figure QM3 is aware of both the T_B and T_C administered clustered topic objects, whereas QM1 is only aware of T_B. QM3 applies both topic definitions to its local topic, and therefore has a different behavor to QM1 for certain topics. For this reason it is important that clustered topics from different clusters do not interfere with each other. Interference can occur when one clustered topic is defined above or below another clustered topic in a different cluster (for example, they have topic strings of `/Sport` and `/Sport/Football`) or even for the same topic string in both. Another form of interference is when administered clustered topic objects are defined with the same object name in different clusters, but for different topic strings.

If such a configuration is made, the delivery of publications to matching subscriptions becomes very dependent on the relative locations of publishers and subscribers with respect to the cluster. For this reason, you cannot rely on such a configuration, and you should change it to remove the interfering topics.

When planning an overlapping cluster topology with publish/subscribe messaging, you can avoid any interference by treating the topic tree and clustered topic object names as if they span all overlapping clusters in the topology.

Integrating multiple publish/subscribe clusters

If there is a requirement for publish/subscribe messaging to span queue managers in different clusters, there are two options available:

- Connect the clusters together through the use of a publish/subscribe hierarchy configuration. See Combining the topic spaces of multiple clusters.
- Create an additional cluster that overlays the existing clusters and includes all queue managers that need to publish or subscribe to a particular topic.

With the latter option, you should consider carefully the size of the cluster and the most effective cluster routing mechanism. See"Designing publish/subscribe clusters" on page 64.

Design considerations for retained publications in publish/subscribe clusters:

There are a few restrictions to consider when designing a publish/subscribe cluster to work with retained publications.

Considerations

Consideration 1: The following cluster queue managers always store the latest version of a retained publication:

- The publisher's queue manager
- In a topic host routed cluster, the topic host (provided there is only one topic host for the topic, as explained in the next section of this article)
- All queue managers with subscriptions matching the topic string of the retained publication

Consideration 2: Queue managers do not receive updated retained publications while they have no subscriptions. Therefore any retained publication stored on a queue manager that no longer subscribes to the topic will become stale.

Consideration 3: On creating any subscription, if there is a local copy of a retained publication for the topic string, the local copy is delivered to the subscription. If you are the first subscriber for any given topic string, a matching retained publication is also delivered from one of the following cluster members:

- In a direct routed cluster, the publisher's queue manager
- In a topic host routed cluster, the topic hosts for the given topic

Delivery of a retained publication from a topic host or publishing queue manager to the subscribing queue manager is asynchronous to the `MQSUB` calls. Therefore, if you use the `MQSUBRQ` call, the latest retained publication might be missed until a subsequent call to `MQSUBRQ`.

Implications

For any publish/subscribe cluster, when a first subscription is made, the local queue manager might be storing a stale copy of a retained publication and this is the copy that is delivered to the new subscription. The existence of a subscription on the local queue manager means that this will resolve the next time the retained publication is updated.

For a topic host routed publish/subscribe cluster, if you configure more than one topic host for a given topic, new subscribers might receive the latest retained publication from a topic host, or they might receive a stale retained publication from another topic host (with the latest having been lost). For topic host routing, it is usual to configure multiple topic hosts for a given topic. However, if you expect applications to make use of retained publications, you should configure only one topic host for each topic.

For any given topic string, you should use only a single publisher, and ensure the publisher always uses the same queue manager. If you do not do this, different retained publications might be active at different queue managers for the same topic, leading to unexpected behavior. Because multiple proxy subscriptions are distributed, multiple retained publications might be received.

If you are still concerned about subscribers using stale publications, consider setting a message expiry when you create each retained publication.

You can use the **CLEAR TOPICSTR** command to remove a retained publication from a publish/subscribe cluster. In certain circumstances you might need to issue the command on multiple members of the publish/subscribe cluster, as described in **CLEAR TOPICSTR** .

Wildcard subscriptions and retained publications

If you are using wildcard subscriptions, the corresponding proxy subscriptions delivered to other members of the publish/subscribe cluster are wildcarded from the topic separator immediately prior to the first wildcard character. See Wildcards and cluster topics.

Therefore the wildcard used might match more topic strings, and more retained publications, than will match the subscribing application.

This increases the amount of storage needed for the retained publications, and you therefore need to ensure that the hosting queue managers have enough storage capacity.

Related information:

Retained publications

Individual proxy subscription forwarding and publish everywhere

REFRESH CLUSTER considerations for publish/subscribe clusters:

Issuing the **REFRESH CLUSTER** command results in the queue manager temporarily discarding locally held information about a cluster, including any cluster topics and their associated proxy subscriptions.

The time taken from issuing the **REFRESH CLUSTER** command to the point that the queue manager regains a full knowledge of the necessary information for clustered publish/subscribe depends on the size of the cluster, the availability, and the responsiveness of the full repository queue managers.

During the refresh processing, disruption to publish/subscribe traffic in a publish/subscribe cluster occurs. For large clusters, use of the **REFRESH CLUSTER** command can disrupt 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. For these reasons, the **REFRESH CLUSTER** command must be used in a publish/subscribe cluster only when under the guidance of your IBM Support Center.

The disruption to the cluster can appear externally as the following symptoms:

- Subscriptions to cluster topics on this queue manager are not receiving publications from publishers that are connected to other queue managers in the cluster.
- Messages that are published to cluster topics on this queue manager are not being propagated to subscriptions on other queue managers.
- Subscriptions to cluster topics on this queue manager created during this period are not consistently sending proxy subscriptions to other members of the cluster.
- Subscriptions to cluster topics on this queue manager deleted during this period are not consistently removing proxy subscriptions from other members of the cluster.
- 10-second pauses, or longer, in message delivery.
- **MQPUT** failures, for example, MQRC_PUBLICATION_FAILURE.

- Publications placed on the dead-letter queue with a reason of MQRC_UNKNOWN_REMOTE_Q_MGR

For these reasons publish/subscribe applications need to be quiesced before issuing the **REFRESH CLUSTER** command.

See also Usage notes for **REFRESH CLUSTER** and "Clustering: Using REFRESH CLUSTER best practices" on page 58.

After a **REFRESH CLUSTER** command is issued on a queue manager in a publish/subscribe cluster, wait until all cluster queue managers and cluster topics have been successfully refreshed, then resynchronize proxy subscriptions as described in Resynchronization of proxy subscriptions. When all proxy subscriptions have been correctly resynchronized, restart your publish/subscribe applications.

If a **REFRESH CLUSTER** command is taking a long time to complete, monitor it by looking at the CURDEPTH of `SYSTEM.CLUSTER.COMMAND.QUEUE`.

Related concepts:

"Clustering: Using REFRESH CLUSTER best practices" on page 58
You use the **REFRESH CLUSTER** command to discard all locally held information about a cluster and rebuild that information from the full repositories in the cluster. You should not need to use this command, except in exceptional circumstances. If you do need to use it, there are special considerations about how you use it. This information is a guide based on testing and feedback from customers.

Routing in publish/subscribe hierarchies

If your distributed queue manager topology is a publish/subscribe hierarchy, and a subscription is made on a queue manager, by default a proxy subscription is created on every queue manager in the hierarchy. Publications received on any queue manager are then routed through the hierarchy to each queue manager that hosts a matching subscription.

For an introduction to how messages are routed between queue managers in publish/subscribe hierarchies and clusters, see Distributed publish/subscribe networks.

When a subscription to a topic is made on a queue manager in a distributed publish/subscribe hierarchy, the queue manager manages the process by which the subscription is propagated to connected queue managers. *Proxy subscriptions* flow to all queue managers in the network. A proxy subscription gives a queue manager the information it needs to forward a publication to those queue managers that host subscriptions for that topic. Each queue manager in a publish/subscribe hierarchy is only aware of its direct relations. Publications put to one queue manager are sent, through its direct relations, to those queue managers with subscriptions. This is illustrated in the following figure, in which *Subscriber 1* registers a subscription for a particular topic on the *Asia* queue manager (1). Proxy subscriptions for this subscription on the *Asia* queue manager are forwarded to all other queue managers in the network (2,3,4).

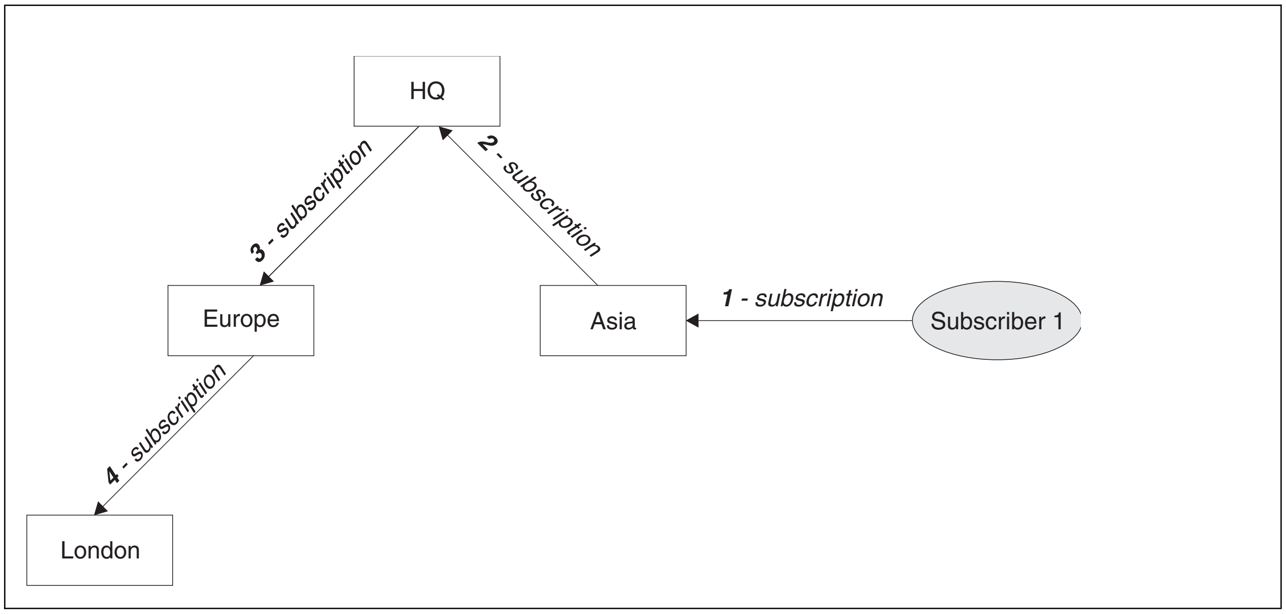


Figure 31. Propagation of subscriptions through a queue manager network

A queue manager consolidates all the subscriptions that are created on it, whether from local applications or from remote queue managers. It creates proxy subscriptions for the topics of the subscriptions with its neighbors, unless a proxy subscription already exists. This is illustrated in the following figure, in which *Subscriber 2* registers a subscription, to the same topic as in Figure 31, on the *HQ* queue manager (5). The subscription for this topic is forwarded to the *Asia* queue manager, so that it is aware that subscriptions exist elsewhere on the network (6). The subscription is not forwarded to the *Europe* queue manager, because a subscription for this topic has already been registered; see step 3 in Figure 31.

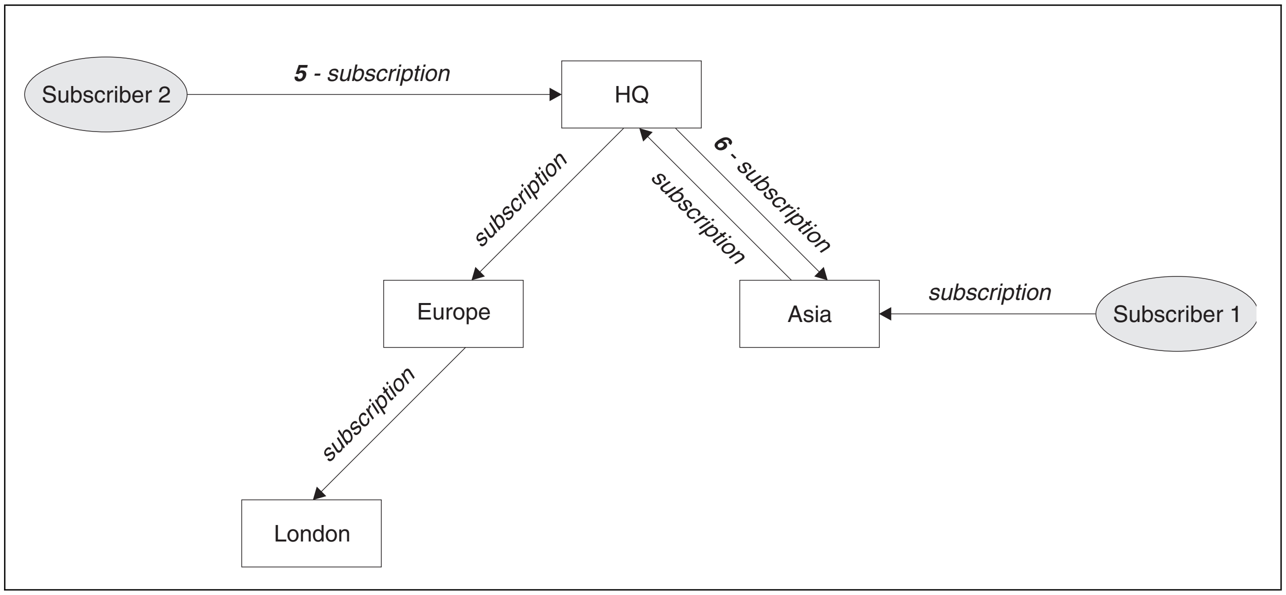


Figure 32. Multiple subscriptions

When an application publishes information to a topic, by default the receiving queue manager forwards it to all queue managers that have valid subscriptions to the topic. It might forward it through one or more intermediate queue managers. This is illustrated in the following figure, in which a publisher sends a publication, on the same topic as in Figure 32, to the *Europe* queue manager (7). A subscription for this topic exists from *HQ* to *Europe*, so the publication is forwarded to the *HQ* queue manager (8). However, no subscription exists from *London* to *Europe* (only from *Europe* to *London*), so the publication is not forwarded to the *London* queue manager. The *HQ* queue manager sends the publication directly to *Subscriber 2* and to the *Asia* queue manager (9). The publication is forwarded to *Subscriber 1* from *Asia* (10).

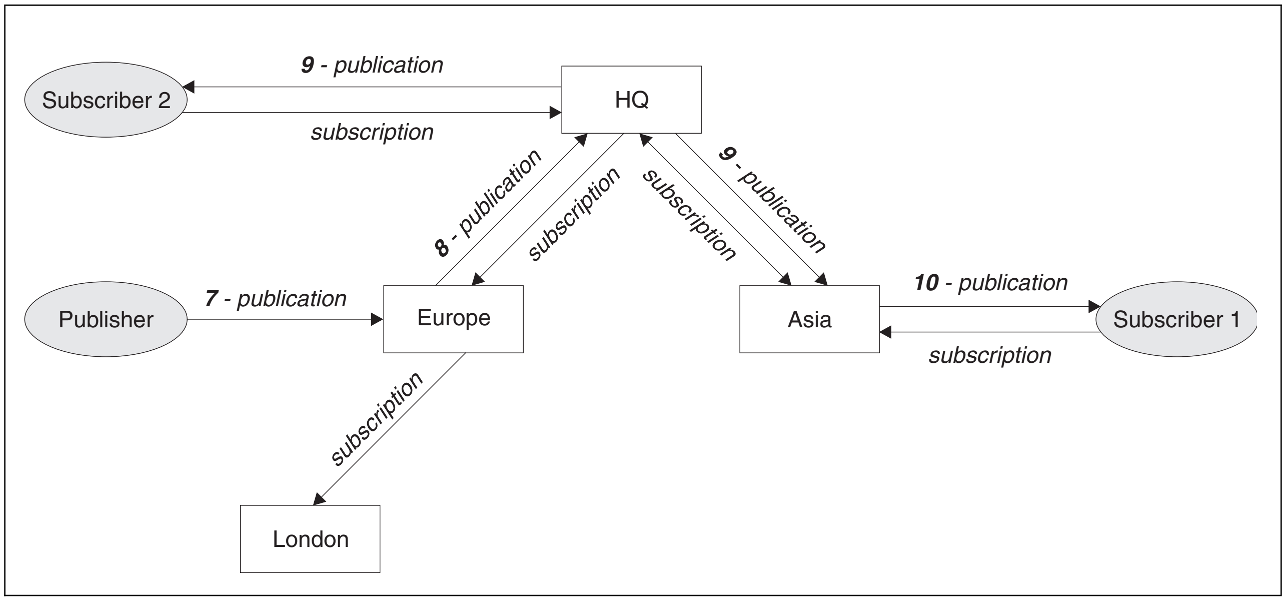


Figure 33. Propagation of publications through a queue manager network

When a queue manager sends any publications or subscriptions to another queue manager, it sets its own user ID in the message. If you are using a publish/subscribe hierarchy, and if the incoming channel is set up to put messages with the authority of the user ID in the message, then you must authorize the user ID of the sending queue manager. See Using default user IDs with a queue manager hierarchy.

Note: If you instead use publish/subscribe clusters, authorization is handled by the cluster.

Summary and additional considerations

A publish/subscribe hierarchy gives you precise control over the relationship between queue managers. After it has been created, it needs little manual intervention to administer. However it also imposes certain constraints upon your system:

- The higher nodes in the hierarchy, especially the root node, must be hosted on robust, highly available, and performant equipment. This is because more publication traffic is expected to flow through these nodes.
- The availability of every non-leaf queue manager in the hierarchy affects the ability of the network to flow messages from publishers to subscribers on other queue managers.
- By default, all topic strings subscribed to are propagated throughout the hierarchy, and publications are propagated only to remote queue managers that have a subscription to the associated topic. Therefore rapid changes to the set of subscriptions can become a limiting factor. You can change this default behavior, and instead have all publications propagated to all queue managers, which removes the need for proxy subscriptions. See Subscription performance in publish/subscribe networks.

  Note: A similar restriction also applies to direct routed clusters.
- Because of the interconnected nature of publish/subscribe queue managers, it takes time for proxy subscriptions to propagate around all nodes in the network. Remote publications do not necessarily start being subscribed to immediately, so early publications might not be sent following a subscription to a new topic string. You can remove the problems caused by the subscription delay by having all

publications propagated to all queue managers, which removes the need for proxy subscriptions. See Subscription performance in publish/subscribe networks.

Note: This restriction also applies to direct routed clusters.

- For a publish/subscribe hierarchy, adding or removing queue managers requires manual configuration to the hierarchy, with careful consideration to the location of those queue managers and their reliance on other queue managers. Unless you are adding or removing queue managers that are at the bottom of the hierarchy, and therefore have no further branches below them, you will also have to configure other queue managers in the hierarchy.

Before you use a publish/subscribe hierarchy as your routing mechanism, explore the alternative approaches detailed in "Direct routing in publish/subscribe clusters" on page 66 and "Topic host routing in publish/subscribe clusters" on page 71.

Distributed publish/subscribe system queues

Four system queues are used by queue managers for publish/subscribe messaging. You need to be aware of their existence only for problem determination and capacity planning purposes.

See Balancing producers and consumers in publish/subscribe networks for guidance on how to monitor these queues.

Table 6. Publish/subscribe system queues on Multiplatforms

System queue	Purpose
SYSTEM.INTER.QMGR.CONTROL	IBM MQ distributed publish/subscribe control queue
SYSTEM.INTER.QMGR.FANREQ	IBM MQ distributed publish/subscribe internal proxy subscription fan-out process input queue
SYSTEM.INTER.QMGR.PUBS	IBM MQ distributed publish/subscribe publications
SYSTEM.HIERARCHY.STATE	IBM MQ distributed publish/subscribe hierarchy relationship state

z/OS On z/OS, you set up the necessary system objects when you create the queue manager, by including the CSQ4INSX, CSQ4INSR and CSQ4INSG samples in the CSQINP2 initialization input data set. For more information, see Task 13: Customize the initialization input data sets.

The attributes of the publish/subscribe system queues are shown in Table 7.

Table 7. Attributes of publish/subscribe system queues

Attribute	Default value
DEFPSIST	Yes
DEFSOPT	EXC
MAXMSGL	Multi On Multiplatforms: The value of the MAXMSGL parameter of the ALTER QMGR command z/OS On z/OS: 104857600 (that is, 100 MB)
MAXDEPTH	999999999
SHARE	N/A
z/OS STGCLASS	This attribute is used only on z/OS platforms

Note: The only queue that contains messages put by applications is `SYSTEM.INTER.QMGR.PUBS`. **`MAXDEPTH`** is set to its maximum value for this queue to allow temporary build up of published messages during outages or times of excessive load. If the queue manager is running on a system where that depth of queue could not be contained, this should be adjusted.

Related information:

Distributed publish/subscribe troubleshooting

Distributed publish/subscribe system queue errors:

Errors can occur when distributed publish/subscribe queue manager queues are unavailable. This affects the propagation of subscription knowledge across the publish/subscribe network, and publication to subscriptions on remote queue managers.

If the fan-out request queue `SYSTEM.INTER.QMGR.FANREQ` is unavailable, the creation of a subscription might generate an error, and error messages will be written to the queue manager error log when proxy subscriptions need to be delivered to directly connected queue managers.

If the hierarchy relationship state queue `SYSTEM.HIERARCHY.STATE` is unavailable, an error message is written to the queue manager error log and the publish/subscribe engine is put into COMPAT mode. To view the publish/subscribe mode, use the command `DISPLAY QMGR PSMODE`.

If any other of the `SYSTEM.INTER.QMGR` queues are unavailable, an error message is written to the queue manager error log and, although function is not disabled, it is likely that publish/subscribe messages will build up on queues on this or remote queue managers.

If the publish/subscribe system queue or required transmission queue to a parent, child or publish/subscribe cluster queue manager is unavailable, the following outcomes occur:

- The publications are not delivered, and a publishing application might receive an error. For details of when the publishing application receives an error, see the following parameters of the **`DEFINE TOPIC`** command: **`PMSGDLV`** , **`NPMSGDLV`** , and **`USEDLQ`** .
- Received inter-queue manager publications are backed out to the input queue, and subsequently re-attempted. If the backout threshold is reached, the undelivered publications are placed on the dead letter queue. The queue manager error log will contain details of the problem.
- An undelivered proxy subscription is backed out to the fanout request queue, and subsequently attempted again. If the backout threshold is reached, the undelivered proxy subscription is not delivered to any connected queue manager, and is placed on the dead letter queue. The queue manager error log will contain details of the problem, including details of any necessary corrective administrative action required.
- Hierarchy relationship protocol messages fail, and the connection status is flagged as `ERROR`. To view the connection status, use the command **`DISPLAY PUBSUB`**.

Related information:
Distributed publish/subscribe troubleshooting

Planning your storage and performance requirements on Multiplatforms

Multi

You must set realistic and achievable storage, and performance goals for your IBM MQ system. Use the links to find out about factors that affect storage and performance on your platform.

The requirements vary depending on the systems that you are using IBM MQ on, and what components you want to use.

For the latest information about supported hardware and software environments, see System Requirements for IBM MQ.

IBM MQ stores queue manager data in the file system. Use the following links to find out about planning and configuring directory structures for use with IBM MQ:
- “Planning file system support on Multiplatforms” on page 104
- “Requirements for shared file systems on Multiplatforms” on page 105
- “Sharing IBM MQ files on Multiplatforms” on page 114
- Linux UNIX “Directory structure on UNIX and Linux systems” on page 117
- Windows “Directory structure on Windows systems” on page 126
- IBM i “Directory structure on IBM i” on page 130

Linux UNIX Use the following links for information about system resources, shared memory, and process priority on UNIX and Linux:
- Linux UNIX “IBM MQ and UNIX System V IPC resources” on page 134
- AIX “Shared memory on AIX” on page 134
- Linux UNIX “IBM MQ and UNIX Process Priority” on page 135

Use the following links for information about log files:
- “Choosing circular or linear logging on Multiplatforms” on page 133
- Calculating the size of the log

Related concepts:

"Planning an IBM MQ architecture" on page 1
When planning your IBM MQ environment, consider the support that IBM MQ provides for single and multiple queue manager architectures, and for point-to-point and publish/subscribe messaging styles. Also plan your resource requirements, and your use of logging and backup facilities.

z/OS "Planning your IBM MQ environment on z/OS" on page 138
When planning your IBM MQ environment, you must consider the resource requirements for data sets, page sets, Db2, Coupling Facilities, and the need for logging, and backup facilities. Use this topic to plan the environment where IBM MQ runs.

Related information:

Hardware and software requirements on UNIX and Linux

Hardware and software requirements on Windows

Disk space requirements on Multiplatforms

Multi

The storage requirements for IBM MQ depend on which components you install, and how much working space you need.

Disk storage is required for the optional components you choose to install, including any prerequisite components they require. The total storage requirement also depends on the number of queues that you use, the number and size of the messages on the queues, and whether the messages are persistent. You also require archiving capacity on disk, tape, or other media, as well as space for your own application programs.

The following table shows the approximate disk space required when you install various combinations of the product on different platforms. (Values are rounded up to the nearest 5 MB, where a MB is 1,048,576 bytes.)

IBM i

Notes:

1. On IBM i you cannot separate the native client from the server. The server figure in the table is for 5724H72*BASE without Java™, together with the English Language Load (2924). There are 22 possible unique language loads.
2. The figure in the table is for the native client 5725A49 *BASE without Java.
3. Java and JMS classes can be added to both server and client bindings. If you want to include these features add 110 MB.
4. Adding samples source to either the client or server adds an extra 10 MB.
5. Adding samples to Java and JMS classes adds an extra 5 MB.

Platform	Client installation [1]	Server installation [2]	IBM MQ MFT installation [3]	Full installation - LTS release [4]	Full installation - CD releases [4]
AIX AIX®	145 MB	190 MB	705 MB	915 MB	915 MB V 9.0.5 1410 MB
HP-UX HP-UX	225 MB	310 MB	1075 MB	1340 MB	N/A
IBM i IBM i	215 MB	450 MB	80 MB	655 MB	N/A
Linux Linux for System x (64 bit)	125 MB	170 MB	575 MB	935 MB	V 9.0.1 1500 MB V 9.0.2 1560 MB
Linux Linux on POWER® Systems - Little Endian	90 MB	125 MB	555 MB	685 MB	V 9.0.1 980 MB V 9.0.2 995 MB V 9.0.3 1000 MB V 9.0.4 1100 MB
Linux Linux for IBM Z	125 MB	160 MB	560 MB	665 MB	V 9.0.1 V 9.0.2 1050 MB V 9.0.3 1100 MB
Solaris Solaris x86-64, AMD64, EM64T, and compatible processors	105 MB [5]	150 MB [5]	695 MB	860 MB	N/A
Solaris Solaris SPARC	105 MB [5]	150 MB [5]	680 MB	820 MB	N/A
Windows Windows (64 bit installation) [6]	445 MB	555 MB	710 MB	1070 MB	V 9.0.1 V 9.0.2 1490 MB V 9.0.3 1310 MB

Usage notes

1. A client installation includes the following components:
   - Runtime
   - Client
2. A server installation includes the following components:
   - Runtime
   - Server
3. A Managed File Transfer installation includes the following components:
   - Managed File Transfer Service, Logger, Agent, Tools, and Base components
   - Runtime
   - Server
   - Java
   - JRE
4. A full installation includes all available components.
5. Solaris On Solaris platforms you must install silently to get this combination of components.
6. Windows Not all the components listed here are installable features on Windows systems; their functionality is sometimes included in other features. See IBM MQ features for Windows systems.

Related information:

IBM MQ components and features

Planning file system support on Multiplatforms

Multi

Queue manager data is stored in the file system. A queue manager makes use of file system locking to prevent multiple instances of a multi-instance queue manager being active at the same time.

Shared file systems

Shared file systems enable multiple systems to access the same physical storage device concurrently. Corruption would occur if multiple systems accessed the same physical storage device directly without some means of enforcing locking and concurrency control. Operating systems provide local file systems with locking and concurrency control for local processes; network file systems provide locking and concurrency control for distributed systems.

Historically, networked file systems have not performed fast enough, or provided sufficient locking and concurrency control, to meet the requirements for logging messages. Today, networked file systems can provide good performance, and implementations of reliable network file system protocols such as *RFC 3530, Network File System (NFS) version 4 protocol*, meet the requirements for logging messages reliably.

Shared file systems and IBM MQ

Queue manager data for a multi-instance queue manager is stored in a shared network file system. On Microsoft UNIX, Linux, and Windows systems, the queue manager's data files and log files must be placed in shared network file system. IBM i On IBM i, journals are used instead of log files, and journals cannot be shared. Multi-instance queue managers on IBM i use journal replication, or switchable journals, to make journals available between different queue manager instances.

Prior to release v7.0.1, IBM MQ does not support queue manager data stored on networked storage accessed as a shared file system. If queue manager data is placed on shared networked storage, then you need to ensure the queue manager data is not accessed by another instance of the queue manager running at the same time.

From v7.0.1 onwards, IBM MQ uses locking to prevent multiple instances of the same multi-instance queue manager being active at the same time. The same locking also ensures that two separate queue managers cannot inadvertently use the same set of queue manager data files. Only one instance of a queue manager can have its lock at a time. Consequently, IBM MQ does support queue manager data stored on networked storage accessed as a shared file system.

Because not all the locking protocols of network file systems are robust, and because a file system might be configured for performance rather than data integrity, you must run the **amqmfsck** command to test whether a network file system will control access to queue manager data and logs correctly. This command applies only UNIX and IBM i systems. On Microsoft Windows, there is only one supported network file system and the **amqmfsck** command is not required.

Related tasks:

"Verifying shared file system behavior on Multiplatforms" on page 107

Run **amqmfsck** to check whether a shared file system on UNIX IBM i and IBM i systems meets the requirements for storing the queue manager data of a multi-instance queue manager. Run the IBM MQ MQI client sample program **amqsfhac** in parallel with **amqmfsck** to demonstrate that a queue manager maintains message integrity during a failure.

Requirements for shared file systems on Multiplatforms

Multi

Shared files systems must provide data write integrity, guaranteed exclusive access to files and release locks on failure to work reliably with IBM MQ.

There are three fundamental requirements that a shared file system must meet:

1. Data write integrity.

   Data write integrity is sometimes called "Write through to disk on flush". The queue manager must be able to synchronize with data being successfully committed to the physical device. In a transactional system, you need to be sure that some writes have been safely committed before continuing with other processing.

   More specifically, IBM MQ on UNIX platforms uses the *O_SYNC* open option and the `fsync()` system call to explicitly force writes to recoverable media, and the write operation is dependent upon these options operating correctly.

   Attention: Linux You should mount the file system with the `async` option, which still supports the option of synchronous writes and gives better performance than the `sync` option.

   Note, however, that if the file system has been exported from Linux, you must still export the file system using the `sync` option.

2. Guaranteed exclusive access to files.

   In order to synchronize multiple queue managers, there needs to be a mechanism for a queue manager to obtain an exclusive lock on a file.

3. Release locks on failure.

   If a queue manager fails, or if there is a communication failure with the file system, files locked by the queue manager need to be unlocked and made available to other processes without waiting for the queue manager to be reconnected to the file system.

   For multi-instance queue managers on Microsoft Windows, the networked storage must be accessed by the Common Internet File System (CIFS) protocol used by Microsoft Windows networks.

   For multi-instance queue managers on other supported platforms, the storage must be accessed by a network file system protocol which is Posix-compliant and supports lease-based locking. Network File

System Version 4 satisfies this requirement. Older file systems, such as Network File System Version 3, which do not have a reliable mechanism to release locks after a failure, must not be used with multi-instance queue managers.

A shared file system must meet these requirements for IBM MQ to operate reliably. If it does not, the queue manager data and logs get corrupted when using the shared file system in a multi-instance queue manager configuration.

You must check whether the shared file system you plan to use meets these requirements. You must also check whether the file system is correctly configured for reliability. Shared file systems sometimes provide configuration options to improve performance at the expense of reliability.

For further information, see Testing and support statement for IBM MQ multi-instance queue managers.

Under normal circumstances IBM MQ operates correctly with attribute caching and it is not necessary to disable caching, for example by setting NOAC on an NFS mount. Attribute caching can cause issues when multiple file system clients are contending for write access to the same file on the file system server, as the cached attributes used by each client might not be the same as those attributes on the server. An example of files accessed in this way are queue manager error logs for a multi-instance queue manager. The queue manager error logs might be written to by both an active and a standby queue manager instance and cached file attributes might cause the error logs to grow larger than expected, before rollover of the files occurs.

To help to check the file system, run the task Verifying shared file system behavior. This task checks if your shared file system meets requirements 2 and 3. You need to verify requirement 1 in your shared file system documentation, or by experimenting with logging data to the disk.

Disk faults can cause errors when writing to disk, which IBM MQ reports as First Failure Data Capture errors. You can run the file system checker for your operating system to check the shared file system for any disk faults. For example, on UNIX the file system checker is called fsck. On Windows platforms the file system checker is called CHKDSK, or SCANDISK.

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.

Verifying shared file system behavior on Multiplatforms: Multi

Run **amqmfsck** to check whether a shared file system on UNIX IBM i and IBM i systems meets the requirements for storing the queue manager data of a multi-instance queue manager. Run the IBM MQ MQI client sample program **amqsfhac** in parallel with **amqmfsck** to demonstrate that a queue manager maintains message integrity during a failure.

Before you begin

You need a server with networked storage, and two other servers connected to it that have IBM MQ installed. You must have administrator (root) authority to configure the file system, and be an IBM MQ Administrator to run **amqmfsck**.

About this task

"Requirements for shared file systems on Multiplatforms" on page 105 describes the file system requirements for using a shared file system with multi-instance queue managers. The IBM MQ technote Testing and support statement for IBM MQ multi-instance queue managers lists the shared file systems that IBM has already tested with. The procedure in this task describes how to test a file system to help you assess whether an unlisted file system maintains data integrity.

Failover of a multi-instance queue manager can be triggered by hardware or software failures, including networking problems which prevent the queue manager writing to its data or log files. Mainly, you are interested in causing failures on the file server. But you must also cause the IBM MQ servers to fail, to test any locks are successfully released. To be confident in a shared file system, test all of the following failures, and any other failures that are specific to your environment:
1. Shutting down the operating system on the file server including syncing the disks.
2. Halting the operating system on the file server without syncing the disks.
3. Pressing the reset button on each of the servers.
4. Pulling the network cable out of each of the servers.
5. Pulling the power cable out of each of the servers.
6. Switching off each of the servers.

Create the directory on the networked storage that you are going to use to share queue manager data and logs. The directory owner must be an IBM MQ Administrator, or in other words, a member of the `mqm` group on UNIX. The user who runs the tests must have IBM MQ Administrator authority.

Use the example of exporting and mounting a file system in Create a multi-instance queue manager on Linux IBM i or Mirrored journal configuration on an ASP using ADDMQMJRN to help you through configuring the file system. Different file systems require different configuration steps. Read the file system documentation.

Procedure

In each of the checks, cause all the failures in the previous list while the file system checker is running. If you intend to run **amqsfhac** at the same time as **amqmfsck**, do the task, "Running **amqsfhac** to test message integrity" on page 112 in parallel with this task.
1. Mount the exported directory on the two IBM MQ servers.

   On the file system server create a shared directory `shared`, and a subdirectory to save the data for multi-instance queue managers, `qmdata`. For an example of setting up a shared directory for multi-instance queue managers on Linux, see Example in Create a multi-instance queue manager on Linux

2. Check basic file system behavior.

   On one IBM MQ server, run the file system checker with no parameters.

```
amqmfsck /shared/qmdata
```

Figure 34. On IBM MQ server 1

3. Check concurrently writing to the same directory from both IBM MQ servers.

   On both IBM MQ servers, run the file system checker at the same time with the `-c` option.

```
amqmfsck -c /shared/qmdata
```

Figure 35. On IBM MQ server 1

```
amqmfsck -c /shared/qmdata
```

Figure 36. On IBM MQ server 2

4. Check waiting for and releasing locks on both IBM MQ servers.

   On both IBM MQ servers run the file system checker at the same time with the `-w` option.

```
amqmfsck -w /shared/qmdata
```

Figure 37. On IBM MQ server 1

```
amqmfsck -w /shared/qmdata
```

Figure 38. On IBM MQ server 2

5. Check for data integrity.
   a. Format the test file.

      Create a large file in the directory being tested. The file is formatted so that the subsequent phases can complete successfully. The file must be large enough that there is sufficient time to interrupt the second phase to simulate the failover. Try the default value of 262144 pages (1 GB). The program automatically reduces this default on slow file systems so that formatting completes in about 60 seconds

```
amqmfsck -f /shared/qmdata
```

The server responds with the following messages:

```
Formatting test file for data integrity test.


Test file formatted with 262144 pages of data.
```

Figure 39. On IBM MQ server 1

   b. Write data into the test file using the file system checker while causing a failure.

      Run the test program on two servers at the same time. Start the test program on the server which is going to experience the failure, then start the test program on the server that is going to survive the failure. Cause the failure you are investigating.

The first test program stops with an error message. The second test program obtains the lock on the test file and writes data into the test file starting where the first test program left off. Let the second test program run to completion.

Table 8. Running the data integrity check on two servers at the same time

IBM MQ server 1	**IBM MQ server 2**
`amqmfsck -a /shared/qmdata`	
`Please start this program on a second machine with the same parameters.`	`amqmfsck -a /shared/qmdata`
	`Waiting for lock...`
`File lock acquired.`	`Waiting for lock...`
`Start a second copy of this program with the same parameters on another server.`	`Waiting for lock...`
`Writing data into test file.`	`Waiting for lock...`
`To increase the effectiveness of the test, interrupt the writing by ending the process, temporarily breaking the network connection to the networked storage, rebooting the server or turning off the power.`	`Waiting for lock...` `Waiting for lock...`
`Turn the power off here.`	

	`File lock acquired.`
	`Reading test file`
	`Checking the integrity of the data read.`
	`Appending data into the test file after data already found.`
	`The test file is full of data. It is ready to be inspected for data integrity.`

The timing of the test depends on the behavior of the file system. For example, it typically takes 30 - 90 seconds for a file system to release the file locks obtained by the first program following a power outage. If you have too little time to introduce the failure before the first test program has filled the file, use the `-x` option of **`amqmfsck`** to delete the test file. Try the test from the start with a larger test file.

c. Verify the integrity of the data in the test file.

```
amqmfsck -i /shared/qmdata
```

The server responds with the following messages:

```
File lock acquired

Reading test file checking the integrity of the data read.

The data read was consistent.

The tests on the directory completed successfully.
```

Figure 40. On IBM MQ server 2

6. Delete the test files.

```
amqmfsck -x /shared/qmdata

Test files deleted.
```

Figure 41. On IBM MQ server 2

The server responds with the message:

```
Test files deleted.
```

Results

The program returns an exit code of zero if the tests complete successfully, and non-zero otherwise.

Examples

The first set of three examples shows the command producing minimal output.

Successful test of basic file locking on one server

```
> amqmfsck /shared/qmdata
The tests on the directory completed successfully.
```

Failed test of basic file locking on one server

```
> amqmfsck /shared/qmdata
AMQ6245: Error Calling 'write()[2]' on file '/shared/qmdata/amqmfsck.lck' error '2'.
```

Successful test of locking on two servers

Table 9. Successful locking on two servers

IBM MQ server 1	IBM MQ server 2

`> amqmfsck -w /shared/qmdata`
`Please start this program on a second`
`machine with the same parameters.`
`Lock acquired.`
`Press Return`
`or terminate the program to release the lock.`	

	`> amqmfsck -w /shared/qmdata`
	`Waiting for lock...`

`[ Return pressed ]`
`Lock released.`	

Table 9. Successful locking on two servers (continued)

IBM MQ server 1	IBM MQ server 2

	`Lock acquired.`
	`The tests on the directory completed successfully`

The second set of three examples shows the same commands using verbose mode.

Successful test of basic file locking on one server

```
> amqmfsck -v /shared/qmdata
System call: stat("/shared/qmdata")'
System call: fd = open("/shared/qmdata/amqmfsck.lck", O_RDWR, 0666)
System call: fchmod(fd, 0666)
System call: fstat(fd)
System call: fcntl(fd, F_SETLK, F_WRLCK)
System call: write(fd)
System call: close(fd)
System call: fd = open("/shared/qmdata/amqmfsck.lck", O_RDWR, 0666)
System call: fcntl(fd, F_SETLK, F_WRLCK)
System call: close(fd)
System call: fd1 = open("/shared/qmdata/amqmfsck.lck", O_RDWR, 0666)
System call: fcntl(fd1, F_SETLK, F_RDLCK)
System call: fd2 = open("/shared/qmdata/amqmfsck.lck", O_RDWR, 0666)
System call: fcntl(fd2, F_SETLK, F_RDLCK)
System call: close(fd2)
System call: write(fd1)
System call: close(fd1)
The tests on the directory completed successfully.
```

Failed test of basic file locking on one server

```
> amqmfsck -v /shared/qmdata
System call: stat("/shared/qmdata")
System call: fd = open("/shared/qmdata/amqmfsck.lck", O_RDWR, 0666)
System call: fchmod(fd, 0666)
System call: fstat(fd)
System call: fcntl(fd, F_SETLK, F_WRLCK)
System call: write(fd)
System call: close(fd)
System call: fd = open("/shared/qmdata/amqmfsck.lck", O_RDWR, 0666)
System call: fcntl(fd, F_SETLK, F_WRLCK)
System call: close(fd)
System call: fd = open("/shared/qmdata/amqmfsck.lck", O_RDWR, 0666)
System call: fcntl(fd, F_SETLK, F_RDLCK)
System call: fdSameFile = open("/shared/qmdata/amqmfsck.lck", O_RDWR, 0666)
System call: fcntl(fdSameFile, F_SETLK, F_RDLCK)
System call: close(fdSameFile)
System call: write(fd)
AMQxxxx: Error calling 'write()[2]' on file '/shared/qmdata/amqmfsck.lck', errno 2
(Permission denied).
```

Successful test of locking on two servers

Table 10. Successful locking on two servers - verbose mode

IBM MQ server 1	**IBM MQ server 2**

`> amqmfsck -wv /shared/qmdata`
`Calling 'stat("/shared/qmdata")'`
`Calling 'fd = open("/shared/qmdata/amqmfsck.lkw",`
`O_EXCL | O_CREAT | O_RDWR, 0666)'`
`Calling 'fchmod(fd, 0666)'`
`Calling 'fstat(fd)'`
`Please start this program on a second`
`machine with the same parameters.`
`Calling 'fcntl(fd, F_SETLK, F_WRLCK)'`
`Lock acquired.`
`Press Return`
`or terminate the program to release the lock.`	

	`> amqmfsck -wv /shared/qmdata`
	`Calling 'stat("/shared/qmdata")'`
	`Calling 'fd = open("/shared/qmdata/amqmfsck.lkw",`
	`O_EXCL | O_CREAT | O_RDWR,0666)'`
	`Calling 'fd = open("/shared/qmdata/amqmfsck.lkw,`
	`O_RDWR, 0666)'`
	`Calling 'fcntl(fd, F_SETLK, F_WRLCK)`
	`'Waiting for lock...`

`[ Return pressed ]`
`Calling 'close(fd)'`
`Lock released.`	

	`Calling 'fcntl(fd, F_SETLK, F_WRLCK)'`
	`Lock acquired.`
	`The tests on the directory completed successfully`

Related information:

amqmfsck (file system check)

Running **amqsfhac** *to test message integrity:*

amqsfhac checks that a queue manager using networked storage maintains data integrity following a failure.

Before you begin

You require four servers for this test. Two servers for the multi-instance queue manager, one for the file system, and one for running **amqsfhac** as a IBM MQ MQI client application.

Follow step 1 on page 107 in Procedure to set up the file system for a multi-instance queue manager.

About this task

Procedure

1. Create a multi-instance queue manager on another server, QM1, using the file system you created in step 1 on page 107 in Procedure.

   See Create a multi-instance queue manager.
2. Start the queue manager on both servers making it highly available.

   On server 1:
   ```
   strmqm -x QM1
   ```
   On server 2:
   ```
   strmqm -x QM1
   ```
3. Set up the client connection to run **amqsfhac**.

   a. Use the procedure in *Verifying an IBM MQ installation* for the platform, or platforms, that your enterprise use to set up a client connection, or the example scripts in Reconnectable client samples.
   b. Modify the client channel to have two IP addresses, corresponding to the two servers running QM1.

      In the example script, modify:

      ```
      DEFINE CHANNEL(CHANNEL1) CHLTYPE(CLNTCONN) TRPTYPE(TCP) +
      CONNAME('LOCALHOST(2345)') QMNAME(QM1) REPLACE
      ```

      To:

      ```
      DEFINE CHANNEL(CHANNEL1) CHLTYPE(CLNTCONN) TRPTYPE(TCP) +
      CONNAME('server1(2345),server2(2345)') QMNAME(QM1) REPLACE
      ```

      Where server1 and server2 are the host names of the two servers, and 2345 is the port that the channel listener is listening on. Usually this defaults to 1414. You can use 1414 with the default listener configuration.
4. Create two local queues on QM1 for the test. Run the following MQSC script:

   ```
   DEFINE QLOCAL(TARGETQ) REPLACE
   DEFINE QLOCAL(SIDEQ) REPLACE
   ```
5. Test the configuration with **amqsfhac**

   ```
   amqsfhac QM1 TARGETQ SIDEQ 2 2 2
   ```
6. Test message integrity while you are testing file system integrity.

   Run **amqsfhac** during step 5 on page 108 of Procedure.

   ```
   amqsfhac QM1 TARGETQ SIDEQ 10 20 0
   ```

   If you stop the active queue manager instance, **amqsfhac** reconnects to the other queue manager instance once it has become active. Restart the stopped queue manager instance again, so that you can reverse the failure in your next test. You will probably need to increase the number of iterations based on experimentation with your environment so that the test program runs for sufficient time for the failover to occur.

Results

An example of running **amqsfhac** in step 6 is shown in Figure 42 on page 114. The test is a success.

If the test detected a problem, the output would report the failure. In some test runs MQRC_CALL_INTERRUPTED might report “Resolving to backed out”. It makes no difference to the result. The outcome depends on whether the write to disk was committed by the networked file storage before or after the failure took place.

```
Sample AMQSFHAC start
qmname = QM1
qname = TARGETQ
sidename = SIDEQ
transize = 10
iterations = 20
verbose = 0
Iteration 0
Iteration 1
Iteration 2
Iteration 3
Iteration 4
Iteration 5
Iteration 6
Resolving MQRC_CALL_INTERRUPTED
MQGET browse side tranid=14 pSideinfo->tranid=14
Resolving to committed
Iteration 7
Iteration 8
Iteration 9
Iteration 10
Iteration 11
Iteration 12
Iteration 13
Iteration 14
Iteration 15
Iteration 16
Iteration 17
Iteration 18
Iteration 19
Sample AMQSFHAC end
```

Figure 42. Output from a successful run of **`amqsfhac`**

Related information:
High availability sample programs

Sharing IBM MQ files on Multiplatforms

Multi

Some IBM MQ files are accessed exclusively by an active queue manager, other files are shared.

IBM MQ files are split into program files and data files. Program files are typically installed locally on each server running IBM MQ. Queue managers share access to data files and directories in the default data directory. They require exclusive access to their own queue manager directory trees contained in each of the `qmgrs` and `log` directories shown in Figure 43 on page 115.

Figure 43 on page 115 is a high-level view of the IBM MQ directory structure. It shows the directories which can be shared between queue managers and made remote. The details vary by platform. The dotted lines indicate configurable paths.

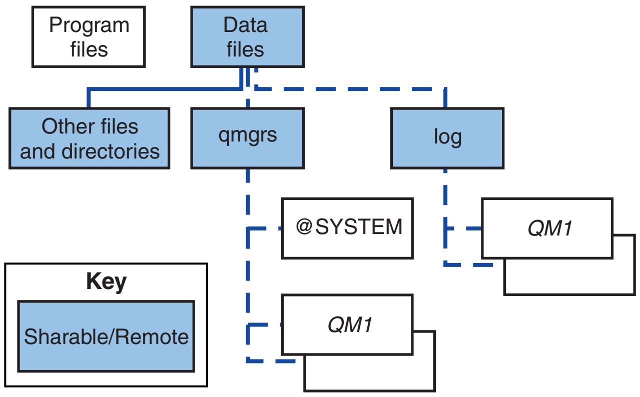


Figure 43. Overall view of IBM MQ directory structure

Program files

The program files directory is typically left in the default location, is local, and shared by all the queue managers on the server.

Data files

The data files directory is typically local in the default location, `/var/mqm` on UNIX and Linux systems and configurable on installation on Windows. It is shared between queue managers. You can make the default location remote, but do not share it between different installations of IBM MQ. The `DefaultPrefix` attribute in the IBM MQ configuration points to this path.

`qmgrs` From v7.0.1, there are two alternative ways to specify the location of queue manager data.

Using `Prefix`

The `Prefix` attribute specifies the location of the `qmgrs` directory. IBM MQ constructs the queue manager directory name from the queue manager name and creates it as a subdirectory of the `qmgrs` directory.

The `Prefix` attribute is located in the QueueManager stanza, and is inherited from the value in the `DefaultPrefix` attribute. By default, for administrative simplicity, queue managers typically share the same `qmgrs` directory.

The `QueueManager` stanza is in the `mqs.ini` file.

If you change the location of the `qmgrs` directory for any queue manager, you must change the value of its `Prefix` attribute.

The `Prefix` attribute for the `QM1` directory in Figure 43 for a UNIX and Linux platform is,

```
Prefix=/var/mqm
```

Using `DataPath`

The `DataPath` attribute specifies the location of the queue manager data directory.

The `DataPath` attribute specifies the complete path, including the name of the queue manager data directory. The `DataPath` attribute is unlike the `Prefix` attribute, which specifies an incomplete path to the queue manager data directory.

The `DataPath` attribute, if it is specified, is located in the QueueManager stanza. If it has been specified, it takes precedence over any value in the `Prefix` attribute.

The `QueueManager` stanza is in the `mqs.ini` file.

If you change the location of the queue manager data directory for any queue manager you must change the value of the `DataPath` attribute.

The `DataPath` attribute for the `QM1` directory in Figure 43 on page 115 for a UNIX or Linux platform is,

```
DataPath=/var/mqm/qmgrs/QM1
```

`log`

The log directory is specified separately for each queue manager in the `Log` stanza in the queue manager configuration. The queue manager configuration is in `qm.ini`.

***`DataPath`*`/`*`QmgrName`*`/@IPCC` subdirectories**

The *`DataPath`*`/`*`QmgrName`*`/@IPCC` subdirectories are in the shared directory path. They are used to construct the directory path for IPC file system objects. They need to distinguish the namespace of a queue manager when a queue manager is shared between systems. Before V7.0.1, a queue manager was only used on one system. One set of subdirectories was sufficient to define the directory path to IPC file system objects, see Figure 44.

```
DataPath/QmgrName/@IPCC/esem
```

Figure 44. Example IPC subdirectory, pre-V7.0.1

In V7.0.1, and later, the IPC file system objects have to be distinguished by system. A subdirectory, for each system the queue manager runs on, is added to the directory path, see Figure 45.

```
DataPath/QmgrName/@IPCC/esem/myHostName/
```

Figure 45. Example IPC subdirectory, V7.0.1 and subsequent releases

`myHostName` is up to the first 20 characters of the host name returned by the operating system. On some systems, the host name might be up to 64 characters in length before truncation. The generated value of *`myHostName`* might cause a problem for two reasons:

1. The first 20 characters are not unique.
2. The host name is generated by a DHCP algorithm that does not always allocate the same host name to a system.

In these cases, set *`myHostName`* using the environment variable, `MQC_IPC_HOST` ; see Figure 46.

```
export MQS_IPC_HOST= myHostName
```

Figure 46. Example: setting `MQC_IPC_HOST`

Other files and directories

Other files and directories, such as the directory containing trace files, and the common error log, are normally shared and kept on the local file system.

Up until v7.0.1, IBM MQ relied upon external management to guarantee queue managers exclusive access to the queue manager data and log files. From v7.0.1 onwards, with support of shared file systems, IBM MQ manages exclusive access to these files using file system locks. A file system lock allows only one instance of a particular queue manager to be active at a time.

When you start the first instance of a particular queue manager it takes ownership of its queue manager directory. If you start a second instance, it can only take ownership if the first instance has stopped. If the first queue manager is still running, the second instance fails to start, and reports the queue manager is running elsewhere. If the first queue manager has stopped, then the second queue manager takes over ownership of the queue manager files and becomes the running queue manager.

You can automate the procedure of the second queue manager taking over from the first. Start the first queue manager with the `strmqm -x` option that permits another queue manager to take over from it. The second queue manager then waits until the queue manager files are unlocked before attempting to take over ownership of the queue manager files, and start.

Directory structure on UNIX and Linux systems

Linux UNIX

The IBM MQ directory structure on UNIX and Linux systems can be mapped to different file systems for easier management, better performance, and better reliability.

Use the flexible directory structure of IBM MQ to take advantage of shared file systems for running multi-instance queue managers.

Use the command **crtmqm** *QM1* to create the directory structure shown in Figure 47 where `R` is the release of the product. It is a typical directory structure for a queue manager created on an IBM MQ system from IBM WebSphere MQ Version 7.0.1 onwards. Some directories, files and .ini attribute settings are omitted for clarity, and another queue manager name might be altered by mangling. The names of the file systems vary on different systems.

In a typical installation, every queue manager that you create points to common `log` and `qmgrs` directories on the local file system. In a multi-instance configuration, the `log` and `qmgrs` directories are on a network file system shared with another installation of IBM MQ.

Figure 47 shows the default configuration for IBM MQ v7.R on AIX where `R` is the release of the product. For examples of alternative multi-instance configurations, see"Example directory configurations on UNIX and Linux systems" on page 121.

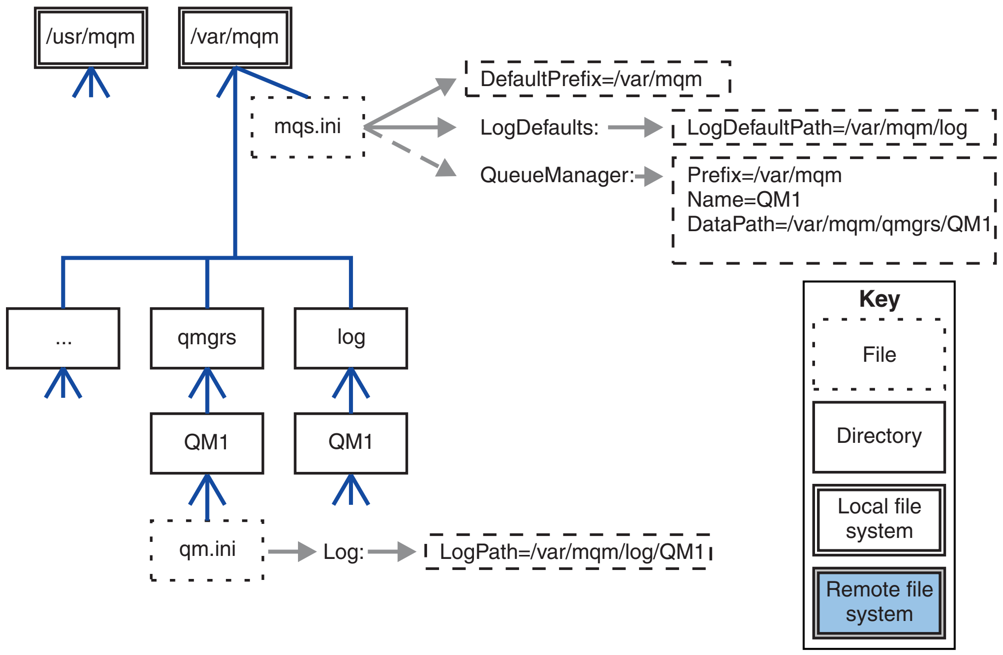


Figure 47. Example default IBM MQ directory structure for UNIX and Linux systems

The product is installed into `/usr/mqm` on AIX and `/opt/mqm` on the other systems, by default. The working directories are installed into the `/var/mqm` directory.

Note: If you created the `/var/mqm` file system prior to installing IBM MQ , ensure that the mqm user has full directory permissions, for example, file mode 755.

Note: The `/var/mqm/errors` directory should be a separate filesystem to prevent FFDCs produced by the queue manager from filling the filesystem that contains `/var/mqm`.

See Creating file systems on UNIX and Linux systems for more information.

The `log` and `qmgrs` directories are shown in their default locations as defined by the default values of the `LogDefaultPath` and `DefaultPrefix` attributes in the `mqs.ini` file. When a queue manager is created, by default the queue manager data directory is created in *`DefaultPrefix`*`/qmgrs`, and the log file directory in *`LogDefaultPath`*`/log`. `LogDefaultPath` and `DefaultPrefix` only effects where queue managers and log files are created by default. The actual location of a queue manager directory is saved in the `mqs.ini` file, and the location of the log file directory is saved in the `qm.ini` file.

The log file directory for a queue manager is defined in the `qm.ini` file in the `LogPath` attribute. Use the `-ld` option on the **`crtmqm`** command to set the `LogPath` attribute for a queue manager; for example, **`crtmqm`** `-ld` *`LogPath`* `QM1`. If you omit the `ld` parameter the value of `LogDefaultPath` is used instead.

The queue manager data directory is defined in the `DataPath` attribute in the `QueueManager` stanza in the `mqs.ini` file. Use the `-md` option on the **`crtmqm`** command to set the `DataPath` for a queue manager; for example, **`crtmqm -`** `md` *`DataPath`* `QM1`. If you omit the `md` parameter the value of the `DefaultPrefix` or `Prefix` attribute is used instead. `Prefix` takes precedence over `DefaultPrefix`.

Typically, create `QM1` specifying both the log and data directories in a single command.

```
 crtmqm
-md DataPath -ld
LogPath QM1
```

You can modify the location of a queue manager log and data directories of an existing queue manager by editing the `DataPath` and `LogPath` attributes in the `qm.ini` file when the queue manager is stopped.

The path to the `errors` directory, like the paths to all the other directories in `/var/mqm`, is not modifiable. However the directories can be mounted on different file systems, or symbolically linked to different directories.

Directory content on UNIX and Linux systems: Linux UNIX

Content of the directories associated with a queue manager.

For information about the location of the product files, see Choosing an installation location

For information about alternative directory configurations, see “Planning file system support on Multiplatforms” on page 104.

In Figure 48 on page 119, the layout is representative of IBM MQ after a queue manager has been in use for some time. The actual structure that you have depends on which operations have occurred on the queue manager.

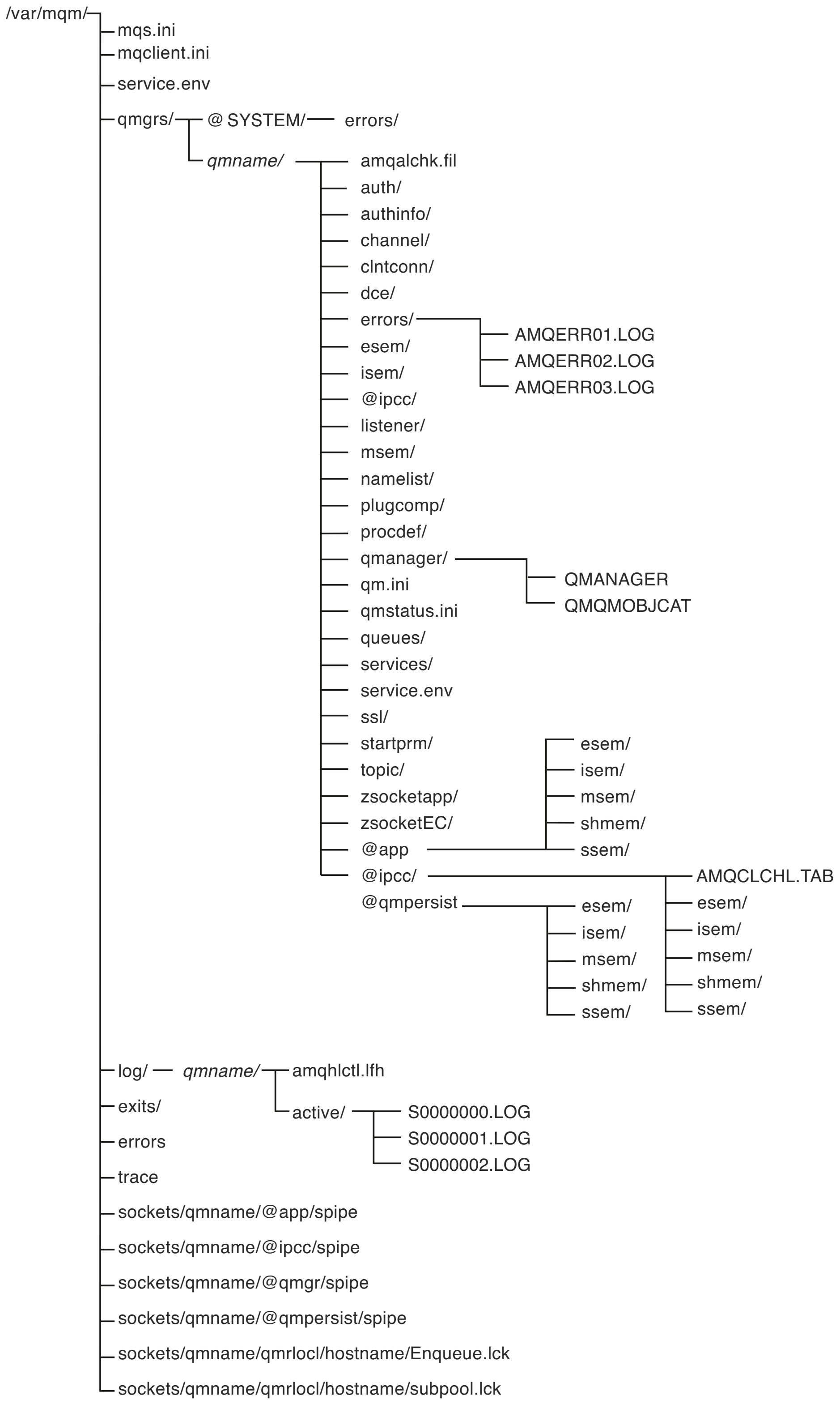


Figure 48. Default directory structure (UNIX) after a queue manager has been started

/var/mqm/

The `/var/mqm` directory contains configuration files and output directories that apply to an IBM MQ installation as a whole, and not to an individual queue manager.

Table 11. Documented content of the `/var/mqm` directory on UNIX

mqs.ini	IBM MQ installation wide configuration file, read when a queue manager starts. File path modifiable using the **AMQ_MQS_INI_LOCATION** environment variable. Ensure this is set and exported in the shell in which the **strmqm** command is run.
mqclient.ini	Default client configuration file read by IBM MQ MQI client programs. File path modifiable using the **MQCLNTCF** environment variable.
service.env	Contains machine scope environment variables for a service process. File path fixed.
errors/	Machine scope error logs, and FFST™ files. Directory path fixed. See also, FFST: IBM MQ for UNIX and Linux systems.
sockets/	Contains information for each queue manager for system use only.
trace/	Trace files. Directory path fixed.
web/	mqweb server directory.
exits/	Default directory containing user channel exit programs. Location modifiable in ApiExit stanzas in the mqs.ini file.
exits64/	

/var/mqm/qmgrs/*qmname***/**

`/var/mqm/qmgrs/`*qmname*`/` contains directories and files for a queue manager. The directory is locked for exclusive access by the active queue manager instance. The directory path is directly modifiable in the `mqs.ini` file, or by using the **md** option of the **crtmqm** command.

Table 12. Documented contents of the `/var/mqm/qmgrs/qmname` directory on UNIX

qm.ini	Queue manager configuration file, read when a queue manager starts.
errors/	Queue manager scope error logs. *qmname* = `@system` contains channel-related messages for an unknown or unavailable queue manager.
@ipcc/ AMQCLCHL.TAB	Default client channel control table, created by the IBM MQ server, and read by IBM MQ MQI client programs. File path modifiable using the **MQCHLLIB** and **MQCHLTAB** environment variables.
qmanager	Queue manager object file: `QMANAGER` Queue manager object catalog: `QMQMOBJCAT`

Table 12. Documented contents of the `/var/mqm/qmgrs/qmname` directory on UNIX (continued)

authinfo/	Each object defined within the queue manager is associated with a file in these directories. The file name approximately matches the definition name; see, Understanding IBM MQ file names.
channel/	
clntconn/	
listener/	
namelist/	
procdef/	
queues/	
services/	
topics/	
...	Other directories used by IBM MQ, such as @ipcc, to be modified only by IBM MQ.

`/var/mqm/log/`*`qmname`*`/`

`/var/mqm/log/`*`qmname`*`/` contains the queue manager log files. The directory is locked for exclusive access by the active queue manager instance. The directory path is modifiable in the `qm.ini` file, or by using the **`ld`** option of the **`crtmqm`** command.

Table 13. Documented contents of the `/var/mqm/log/qmname` directory on UNIX

amqhlctl.lfh	Log control file.
active/	This directory contains the log files numbered S0000000.LOG, S0000001.LOG, S0000002.LOG, and so on.

`opt/mqm`

`opt/mqm` is, by default, the installation directory on most platforms. See "Disk space requirements on Multiplatforms" on page 102 for more information on the amount of space that you need for the installation directory on the platform, or platforms, that your enterprise uses.

Example directory configurations on UNIX and Linux systems: ▶ Linux ▶ UNIX

Examples of alternative file system configurations on UNIX and Linux systems.

You can customize the IBM MQ directory structure in various ways to achieve a number of different objectives.

- Place the `qmgrs` and `log` directories on remote shared file systems to configure a multi-instance queue manager.
- Use separate file systems for the data and log directories, and allocate the directories to different disks, to improve performance by reducing I/O contention.
- Use faster storage devices for directories that have a greater effect on performance. Physical device latency is frequently a more important factor in the performance of persistent messaging than whether a device is mounted locally or remotely. The following list shows which directories are most and least performance sensitive.
  1. `log`
  2. `qmgrs`
  3. Other directories, including `/usr/mqm`
- Create the `qmgrs` and `log` directories on file systems that are allocated to storage with good resilience, such as a redundant disk array, for example.

- It is better to store the common error logs in `var/mqm/errors`, locally, rather than on a network file system, so that error relating to the network file system can be logged.

Figure 49 is a template from which alternative IBM MQ directory structures are derived. In the template, dotted lines represent paths that are configurable. In the examples, the dotted lines are replaced by solid lines that correspond to the configuration information stored in the AMQ_MQS_INI_LOCATION environment variable, and in the `mqs.ini` and `qm.ini` files.

Note: The path information is shown as it appears in the `mqs.ini` or `qm.ini` files. If you supply path parameters in the **`crtmqm`** command, omit the name of the queue manager directory: the queue manager name is added to the path by IBM MQ.

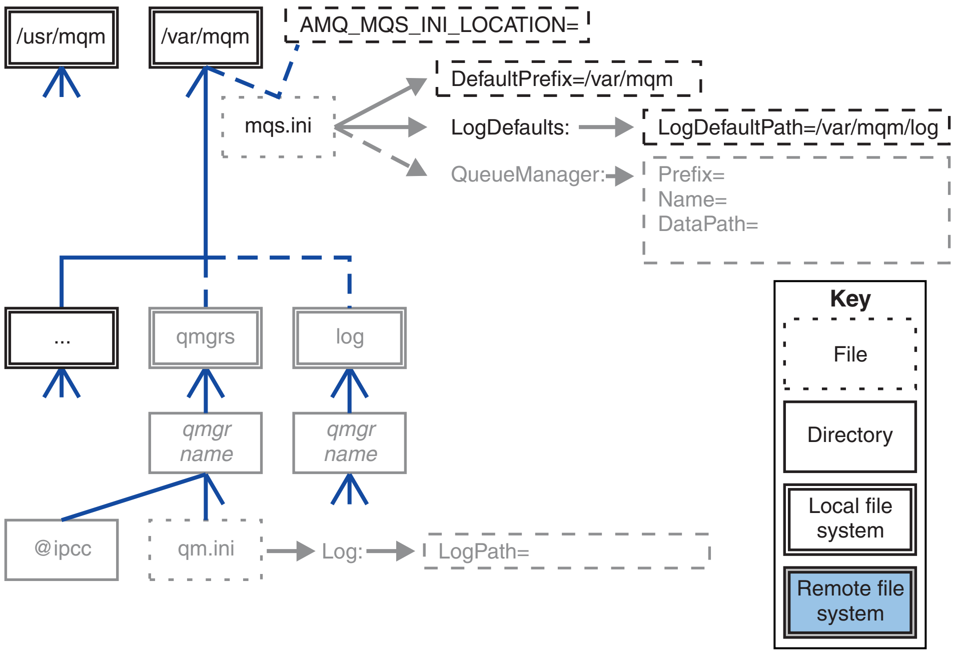


Figure 49. Directory structure pattern template

Examples of configured directory structures follow. The first example shows a typical default directory structure for IBM WebSphere MQ Version 7.0.1 created by issuing the **`crtmqm`** *QM1* command. The second example shows how a typical directory structure appears for a queue manager created using an IBM MQ release earlier than Version 7.0.1. The directory structure does not change.

Queue managers newly created in Version 7.0.1 have a different configuration file from earlier releases of Version 7. If you need to remove the Version 7.0.1 fix pack to revert to Version 7.0.0, Fix Pack 2, you need to re-create the configuration files. You might only need to use the Prefix attribute to define the path to the new queue manager data directory, or you might need to move the queue manager data directory and log directories to a different location. The safest way to reconfigure the queue manager is to save the queue manager data and log directories, delete and re-create the queue manager, and then replace the data and log directories in their new location, with the ones that have been saved.

Typical directory structure for Version 7.0.1 onwards

Figure 50 on page 123 is the default directory structure created in Version 7.0.1 by issuing the command **`crtmqm`** *QM1*.

The mqs.ini file has a stanza for the QM1 queue manager, created by referring to the value of DefaultPrefix. The Log stanza in the qm.ini file has a value for LogPath, set by reference to LogDefaultPath in mqs.ini.

Use the optional **crtmqm** parameters to override the default values of DataPath and LogPath.

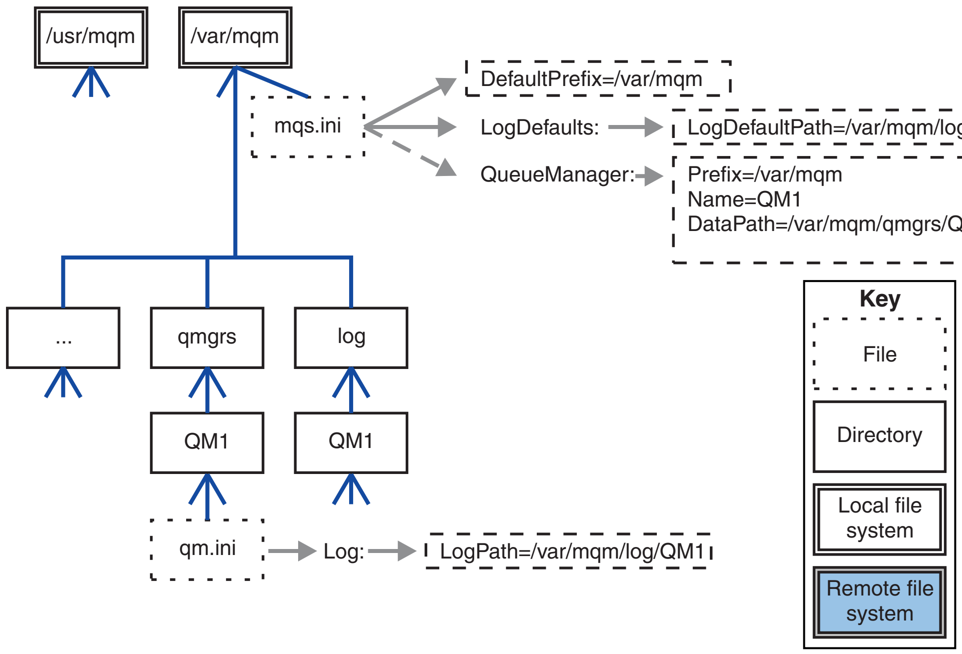


Figure 50. Example default IBM MQ directory structure for UNIX and Linux systems

Typical directory structure for releases earlier than Version 7.0.1

The DataPath attribute did not exist before Version 7.0.1; the attribute is not present in the mqs.ini file. The location of the qmgrs directory was configured using the Prefix attribute. The location of individual directories could be configured by using symbolic links to point to different file system locations.

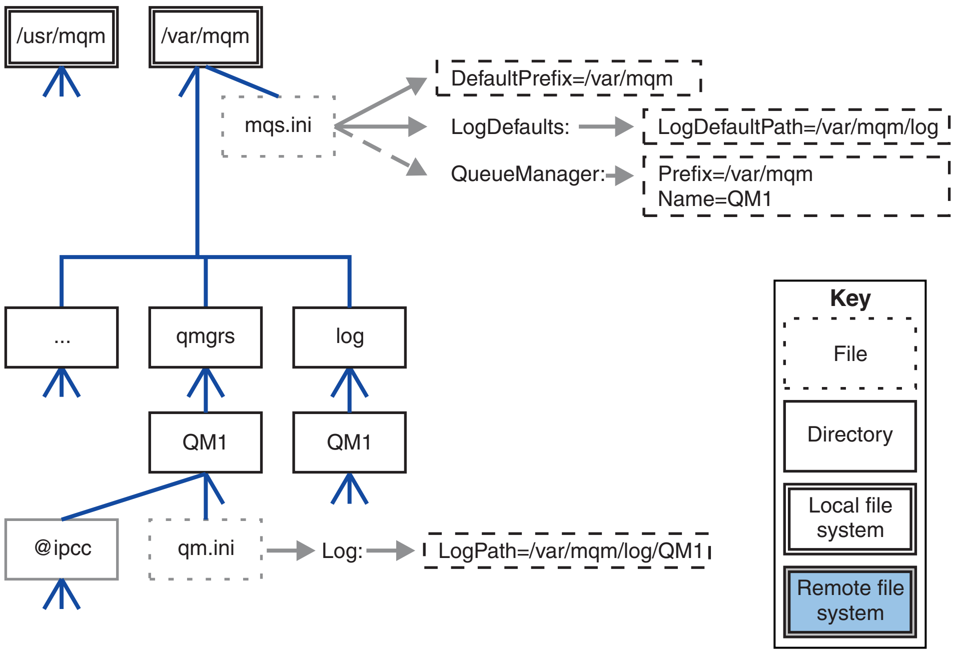


Figure 51. Typical directory structure for releases earlier than Version 7.0.1

Share default `qmgrs` and `log` directories (Version 7.0.1 onwards)

An alternative to "Share everything (Version 7.0.1 onwards)" on page 125, is to share the `qmgrs` and `log` directories separately (Figure 52). In this configuration, there is no need to set `AMQ_MQS_INI_LOCATION` as the default mqs.ini is stored in the local `/var/mqm` file system. The files and directories, such as `mqclient.ini` and `mqserver.ini` are also not shared.

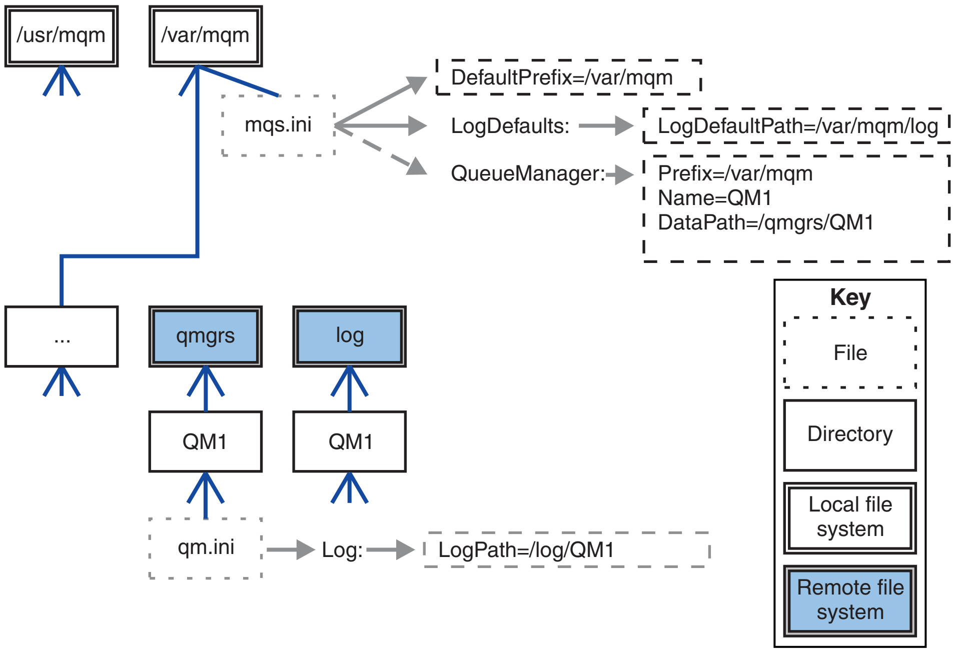


Figure 52. Share `qmgrs` and `log` directories

Share named `qmgrs` and `log` directories (Version 7.0.1 onwards)

The configuration in Figure 53 places the `log` and `qmgrs` in a common named remote shared file system called `/ha`. The same physical configuration can be created in two different ways.

1. Set `LogDefaultPath=/ha` and then run the command, **`crtmqm`** - `md` */ha/qmgrs* `QM1`. The result is exactly as illustrated in Figure 53.
2. Leave the default paths unchanged and then run the command, **`crtmqm`** - `ld` */ha/log* - `md` */ha/qmgrs* `QM1`.

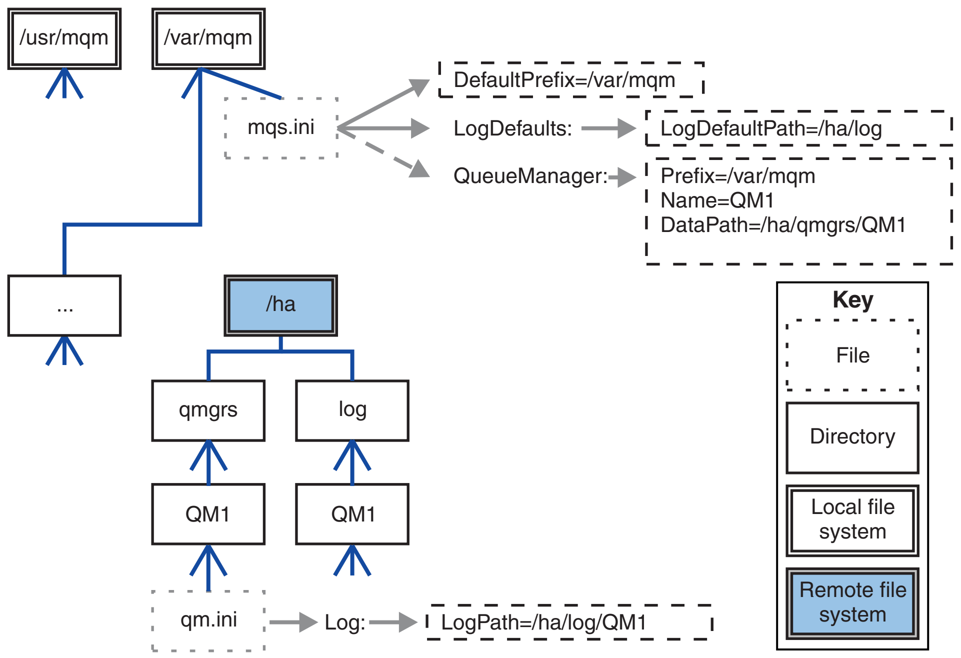


Figure 53. Share named `qmgrs` and `log` directories

Share everything (Version 7.0.1 onwards)

Figure 54 on page 126 is a simple configuration for system with fast networked file storage.

Mount `/var/mqm` as a remote shared file system. By default, when you start QM1, it looks for `/var/mqm`, finds it on the shared file system, and reads the `mqs.ini` file in `/var/mqm`. Rather than use the single `/var/mqm/mqs.ini` file for queue managers on all your servers, you can set the `AMQ_MQS_INI_LOCATION` environment variable on each server to point to different `mqs.ini` files.

Note: The contents of the generic error file in `/var/mqm/errors/` are shared between queue managers on different servers.

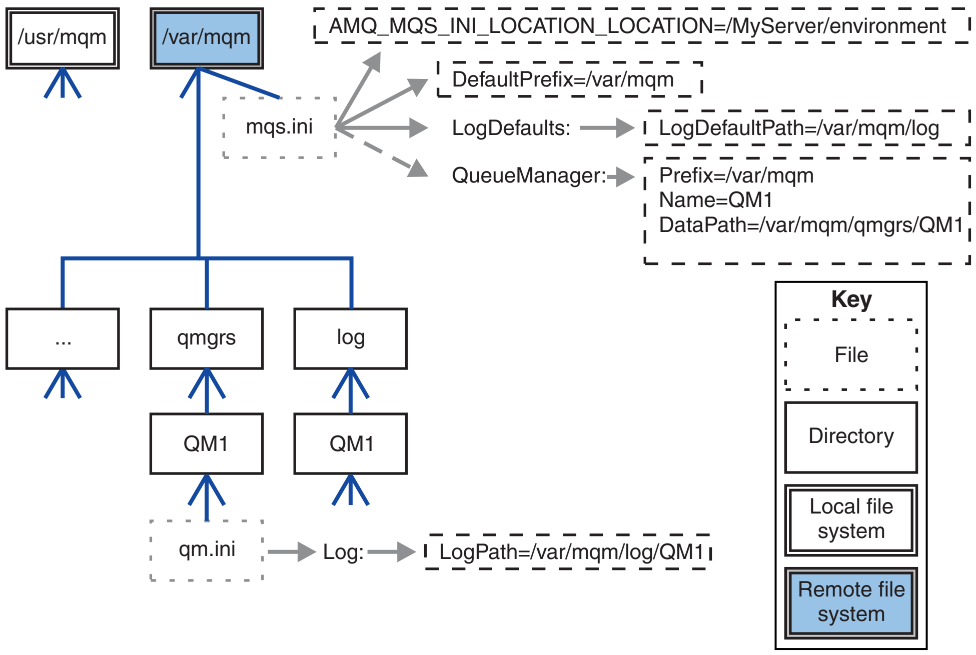


Figure 54. Share everything

Note that you cannot use this for multi-instance queue managers. The reason is that it is necessary for each host in a multi-instance queue manager to have its own local copy of `/var/mqm` to keep track of local data, such as semaphores and shared memory. These entities cannot be shared across hosts.

Directory structure on Windows systems

Windows

How to find queue manager configuration information and directories on Windows.

The default directories for IBM MQ for Windows installation are:

Program directory
`C:\Program Files\IBM\MQ`

Data directory
`C:\ProgramData\IBM\MQ`

Important: Windows For Windows installations, the directories are as stated, unless there is a previous installation of the product that still contains registry entries or queue managers, or both. In this situation, the new installation uses the old data directory location. For more information, see Program and data directory locations.

If you want to know which installation directory and which data directory is being used, run the dspmqver command.

The installation directory is listed in the **`InstPath`** field and the data directory is listed in the **`DataPath`** field.

Running the **`dspmqver`** command displays, for example, the following information:

```
>dspmqver
Name:        IBM MQ
Version:     9.0.0.0
Level:       p900-L160512.4
BuildType:   IKAP - (Production)
Platform:    IBM MQ for Windows (x64 platform)
Mode:        64-bit
O/S:         Windows 7 Professional x64 Edition, Build 7601: SP1
InstName:    Installation1
InstDesc:
Primary:     Yes
InstPath:    C:\Program Files\IBM\MQ
DataPath:    C:\ProgramData\IBM\MQ
MaxCmdLevel: 900
LicenseType: Production
```

Multi-instance queue managers

To configure a multi-instance queue manager, the log and data directories must be placed on networked storage, preferably on a different server to any of the servers that are running instances of the queue manager.

Two parameters are provided on the **crtmqm** command, **-md** and **-ld**, to make it easier specify the location of the queue manager data and log directories. The effect of specifying the **-md** parameter is fourfold:

1. The `mqs.ini` stanza `QueueManager\`*`QmgrName`* contains a new variable, *DataPath*, which points to the queue manager data directory. Unlike the *Prefix* variable, the path includes the name of the queue manager directory.
2. The queue manager configuration information stored in the mqs.ini file is reduced to *Name*, *Prefix*, *Directory* and *DataPath*.

Directory content: Windows

Lists the location and content of IBM MQ directories.

An IBM MQ configuration has three main sets of files and directories:

1. Executable, and other read-only files that are only updated when maintenance is applied. For example:
   - The readme file
   - The IBM MQ Explorer plug-in and help files
   - License files

   These files are described in Table 14 on page 128.
2. Potentially modifiable files and directories that are not specific to a particular queue manager. These files and directories are described in Table 15 on page 128.
3. Files and directories that are specific to each queue manager on a server. These files and directories are described in Table 16 on page 129.

Resource directories and files

The resource directories and files contain all the executable code and resources to run a queue manager. The variable, *FilePath*, in the installation specific IBM MQ configuration registry key, contains the path to the resource directories.

Table 14. Directories and files in the `FilePath` directory

File path	Contents
FilePath\bin	Commands and DLLs
FilePath\bin64	Commands and DLLs (64 bit)
FilePath\conv	Data conversion tables
FilePath\doc	Wizard help files
FilePath\MQExplorer	Explorer and Explorer help Eclipse plug-ins
FilePath\gskit8	Global security kit
FilePath\java	Java resources, including JRE
FilePath\licenses	License information
FilePath\Non_IBM_License	License information
FilePath\properties	Used internally
FilePath\Tivoli	
FilePath\tools	Development resources and samples
FilePath\web	Described in IBM MQ Console and REST API installation component file structure for non-editable files.
FilePath\Uninst	Used internally
FilePath\README.TXT	Readme file

Directories not specific to a queue manager

Some directories contain files, such as trace files and error logs, that are not specific to a specific queue manager. The *DefaultPrefix* variable contains the path to these directories. *DefaultPrefix* is part of the `AllQueueManagers` stanza.

Table 15. Directories and files in `DefaultPrefix` directory

File path	Contents
DefaultPrefix\config	Used internally
DefaultPrefix\conv	V 9.0.0 `ccsid_part2.tbl` and`ccsid.tbl` data conversion control file, described in Data conversion
DefaultPrefix\errors	Non queue manager error logs, `AMQERR` *nn*`.LOG`
DefaultPrefix\exits	Channel exit programs
DefaultPrefix\exits64	Channel exit programs (64 bit)
DefaultPrefix\ipc	Not used
DefaultPrefix\qmgrs	Described in Table 16 on page 129
DefaultPrefix\trace	Trace files
DefaultPrefix\web	Described in IBM MQ Console and REST API installation component file structure for user editable files
DefaultPrefix\amqmjpse.txt	Used internally

Queue manager directories

When you create a queue manager, a new set of directories, specific to the queue manager, is created.

If you create a queue manager with the **-md** *filepath* parameter, the path is stored in the *DataPath* variable in the queue manager stanza of the mqs.ini file. If you create a queue manager without setting the **-md** *filepath* parameter, the queue manager directories are created in the path stored in *DefaultPrefix*, and the path is copied into the *Prefix* variable in the queue manager stanza of the mqs.ini file.

Table 16. Directories and files in `DataPath` and `Prefix\qmgrs\QmgrName` directories

File path	Contents
DataPath\@ipcc	Default location for AMQCLCHL.TAB, the client connection table.
DataPath\authinfo	Used internally.
DataPath\channel	
DataPath\clntconn	
DataPath\errors	Error logs, AMQERR *nn*.LOG
DataPath\listener	Used internally.
DataPath\namelist	
DataPath\plugcomp	
DataPath\procdef	
DataPath\qmanager	
DataPath\queues	
DataPath\services	
DataPath\ssl	
DataPath\startprm	
DataPath\topic	
DataPath\active	
DataPath\active.dat	
DataPath\amqalchk.fil	
DataPath\master	
DataPath\master.dat	
DataPath\qm.ini	Queue manager configuration
DataPath\qmstatus.ini	Queue manager status
Prefix\qmgrs\QmgrName	Used internally
Prefix\qmgrs\@SYSTEM	Not used
Prefix\qmgrs\@SYSTEM\errors	

Directory structure on IBM i

▶ IBM i

A description of the IFS is given, and the IBM MQ IFS directory structure is described for server, client, and Java.

The integrated file system (IFS) is a part of IBM i that supports stream input/output and storage management similar to personal computer, UNIX and Linux operating systems, while providing an integrating structure over all information stored in the server.

On IBM i directory names begin with the character & (ampersand) instead of the character @ (at). For example, @system on IBM i is &system.

IFS root file system for IBM MQ server

When you install IBM MQ Server for IBM i, the following directories are created in the IFS root file system.

ProdData:

Overview

```
QIBM   '-- ProdData
          '-- mqm
           '-- doc
           '-- inc
           '-- lib
           '-- samp
           '-- licenses
          '-- LicenseDoc
           '-- 5724H72_V8R0M0
```

/QIBM/ProdData/mqm
Subdirectories below this contain all the product data, for example, C++ classes, trace format files, and license files. Data in this directory is deleted and replaced each time the product is installed.

/QIBM/ProdData/mqm/doc
A Command Reference for the CL commands is provided in HTML format and installed here.

/QIBM/ProdData/mqm/inc
The header files for compiling your C or C++ programs.

/QIBM/ProdData/mqm/lib
Auxiliary files used by MQ.

/QIBM/ProdData/mqm/samp
Further samples.

/QIBM/ProdData/mqm/licenses
License files. The two files for each language are named like LA_ *xx* and LI_ *xx* where *xx* is the 2 character language identifier for each language supplied.

Also the following directory stores license agreements files:

/QIBM/ProdData/LicenseDoc/5724H72_V8R0M0
License files. The files are named like 5724H72_V8R0M0_ *xx* where *xx* is the 2 or 5 character language identifier for each language supplied.

UserData:

Overview

```
QIBM   '-- UserData
           '-- mqm
            '-- errors
            '-- trace
            '-- qmgrs
            '-- &system
            '-- qmgrname1
            '-- qmgrname2
            '-- and so on
```

/QIBM/UserData/mqm
Subdirectories below this contain all user data relating to queue managers.

When you install the product, an mqs.ini file is created in directory /QIBM/UserData/mqm/ (unless it is already there from a previous installation).

When you create a queue manager, a qm.ini file is created in the directory /QIBM/UserData/mqm/qmgrs/ *QMGRNAME* / (where *QMGRNAME* is the name of the queue manager).

Data in the directories is retained when the product is deleted.

IFS root file system for IBM MQ MQI client

When you install IBM MQ MQI client for IBM i, the following directories created in the IFS root file system:

ProdData:

Overview

```
QIBM   '-- ProdData
           '-- mqm
            '-- lib
```

/QIBM/ProdData/mqm
Subdirectories below this directory contain all the product data. Data in this directory is deleted and replaced each time the product is replaced.

UserData:

Overview

```
QIBM   '-- UserData
           '-- mqm
            '-- errors
            '-- trace
```

/QIBM/UserData/mqm
Subdirectories below this directory contain all user data.

IFS root file system for IBM MQ Java

When you install IBM MQ Java on IBM i, the following directories are created in the IFS root file system:

ProdData:

Overview

```
QIBM   '-- ProdData
            '-- mqm
             '-- java
             '--samples
             '-- bin
             '-- lib
```

/QIBM/ProdData/mqm/java
Subdirectories below this contain all the product data, including Java classes. Data in this directory is deleted and replaced each time the product is replaced.

/QIBM/ProdData/mqm/java/samples
Subdirectories below this contain all the sample Java classes and data.

Libraries created by server and client installations

Installation of the IBM MQ server or client creates the following libraries:

- `QMQM`

  The product library.
- `QMQMSAMP`

  The samples library (if you choose to install the samples).
- `QMxxxx`

  Server only.

  Each time that you create a queue manager, IBM MQ automatically creates an associated library, with a name like `QMxxxx` where `xxxx` is derived from the queue manager name. This library contains objects specific to the queue manager, including journals and associated receivers. By default the name of this library is derived from the name of the queue manager prefixed with the characters QM. For example, for a queue manager called TEST, the library would be called QMTEST.

  Note: When you create a queue manager, you can specify the name of its library if you want to. For example:

  ```
  CRTMQM MQMNAME(TEST) MQMLIB(TESTLIB)
  ```

  You can use the WRKLIB command to list all the libraries that IBM MQ for IBM i has created. Against the queue manager libraries, you will see the text `QMGR: QMGRNAME`. The format of the command is:

  ```
  WRKLIB LIB(QM*)
  ```

  These queue manager-associated libraries are retained when the product is deleted.

Choosing circular or linear logging on Multiplatforms

Multi V 9.0.1

In IBM MQ, you can choose circular or linear logging. The following information gives you an overview of both types.

Advantages of circular logging

The main advantages of circular logging are that circular logging is:

- Easier to administer.

  Once you have configured circular logging correctly for your workload, no further administration is needed. Whereas, for linear logging, media images need to be recorded and log extents that are not required any more need to be archived or deleted.
- Better performing

  Circular logging performs better than linear logging, because circular logging is able to reuse log extents that have already been formatted. Whereas linear logging has to allocate new log extents and format them.

See Managing logs for further information.

Advantages of linear logging

The principal advantage of linear logging is that linear logging provides protection against more failures.

Neither circular nor linear logging protect against a corrupted or deleted log, or messages or queues that have been deleted by applications or the administrator.

Linear logging (but not circular) enables damaged objects to be recovered. So, linear logging provides protection against queue files being corrupted or deleted, as these damaged queues can be recovered from a linear log.

Both circular and linear protect against power loss and communications failure as described in Recovering from power loss or communication failures.

Other considerations

Whether you choose linear or circular depends on how much redundancy you require.

There is a cost to choosing more redundancy, that is linear logging, caused by the performance cost and the administration cost.

See Types of logging for more information.

Shared memory on AIX

AIX

If certain application types fail to connect because of an AIX memory limitation, in most cases this can be resolved by setting the environment variable EXTSHM=ON.

Some 32-bit processes on AIX might encounter an operating system limitation that affects their ability to connect to IBM MQ queue managers. Every standard connection to IBM MQ uses shared memory, but unlike other UNIX and Linux platforms, AIX allows 32-bit processes to attach only 11 shared memory sets.

Most 32-bit processes will not encounter this limit, but applications with high memory requirements might fail to connect to IBM MQ with reason code 2102: MQRC_RESOURCE_PROBLEM. The following application types might see this error:

- Programs running in 32-bit Java virtual machines
- Programs using the large or very large memory models
- Programs connecting to many queue managers or databases
- Programs that attach to shared memory sets on their own

AIX offers an extended shared memory feature for 32-bit processes that allows them to attach more shared memory. To run an application with this feature, export the environment variable EXTSHM=ON before starting your queue managers and your program. The EXTSHM=ON feature prevents this error in most cases, but it is incompatible with programs that use the SHM_SIZE option of the shmctl function.

IBM MQ MQI client applications and all 64-bit processes are unaffected by this limitation. They can connect to IBM MQ queue managers regardless of whether EXTSHM has been set.

IBM MQ and UNIX System V IPC resources

Linux UNIX

A queue manager uses some IPC resources. Use **`ipcs -a`** to find out what resources are being used.

This information applies to IBM MQ running on UNIX and Linux systems only.

IBM MQ uses System V interprocess communication (IPC) resources (*semaphores* and *shared memory segments*) to store and pass data between system components. These resources are used by queue manager processes and applications that connect to the queue manager. IBM MQ MQI clients do not use IPC resources, except for IBM MQ trace control. Use the UNIX command **`ipcs -a`** to get full information on the number and size of the IPC resources currently in use on the machine.

IBM MQ and UNIX Process Priority

Linux UNIX

Good practices when setting process priority *nice* values.

This information applies to IBM MQ running on UNIX and Linux systems only.

If you run a process in the background, that process can be given a higher *nice* value (and hence lower priority) by the invoking shell. This might have general IBM MQ performance implications. In highly-stressed situations, if there are many ready-to-run threads at a higher priority and some at a lower priority, operating system scheduling characteristics can deprive the lower priority threads of processor time.

It is good practice that independently started processes associated with queue managers, such as **runmqlsr**, have the same *nice* values as the queue manager they are associated with. Ensure the shell does not assign a higher *nice* value to these background processes. For example, in ksh, use the setting "`set +o bgnice`" to stop ksh from raising the *nice* value of background processes. You can verify the *nice* values of running processes by examining the *NI* column of a "ps -efl" listing.

Also, start IBM MQ application processes with the same *nice* value as the queue manager. If they run with different *nice* values, an application thread might block a queue manager thread, or vice versa, causing performance to degrade.

Planning your IBM MQ client environment on HP Integrity NonStop Server

NSS Client

When planning the environment in which the IBM MQ client for HP Integrity NonStop Server runs, you must consider the HP Integrity NonStop Server environment, and HP NonStop TMF.

First, familiarize yourself with the basic IBM MQ client for HP Integrity NonStop Server concepts. See the topics in IBM MQ client for HP Integrity NonStop Server technical overview.

Preparing the HP Integrity NonStop Server environment

NSS Client

Before installation, the environment must be prepared depending on whether the installation is to be verified immediately or not.

For the installation, you require the following items:
- A user ID that meets the requirements. For details about user ID requirements, see Setting up the user and group on HP Integrity NonStop Server .
- Verified locations in the OSS and Guardian file systems that can be for the installation files.
- An operational OSS shell and OSS file system. You can verify the file system by doing the following tasks:
  - Log on to the OSS environment (shell). Ensure that you have write access to the OSS installation root directory you intend to use.
  - Log on to the TACL environment using the user ID in the MQM group. Verify that the volume you intend to use meets the requirements and is accessible to you, and that the subvolume does not exist.

  You can login to both OSS or TACL using either an alias, if you have one, or your full principal.

If you intend to proceed immediately to verify that the installation is usable, you might also need the following optional items:

- An operational and accessible Local Sockets subsystem in the OSS environment.
- An operational TCP/IP subsystem.

If you intend to use TMF coordinated global units of work, you will need the following items:

- An operational TMF subsystem.
- An operational Pathway (TS/MP) subsystem.

Work with your systems administrator if you are in any doubt about the status of these critical subsystems.

IBM MQ and HP NonStop TMF

NSS Client

IBM MQ client for HP Integrity NonStop Server can participate in HP NonStop Transaction Management Facility (HP NonStop TMF) coordinated units of work. Coordinating transactions with HP NonStop TMF is only supported where the queue manager is at IBM WebSphere MQ Version 7.1 or later.

The IBM MQ provided TMF/Gateway converts transactions from TMF coordination into eXtended Architecture (XA) transaction coordination to communicate with the remote queue manager. The IBM MQ provided TMF/Gateway is the bridge between TMF and queue manager transactions, using the services provided by HP NonStop TMF, and has been designed to run in a Pathway environment.

HP NonStop TMF software provides transaction protection and database consistency in demanding environments. For more information about HP NonStop TMF, see HP NonStop TMF Introduction.

For information about how to configure the IBM MQ provided TMF/Gateway, see Configuring IBM MQ client for HP Integrity NonStop Server.

Using HP NonStop TMF

NSS Client

The HP NonStop Transaction Management Facility (TMF) is the native transaction manager on HP Integrity NonStop Server and is integrated with the file system and the relational database managers, SQL/MP, and SQL/MX.

IBM MQ client for HP Integrity NonStop Server can use TMF to coordinate global units of work.

To coordinate global units of work, TMF acts as the transaction manager, and an application must use the API provided by TMF to start, commit, and back out global units of work. An application starts a global unit of work by calling BEGINTRANSACTION, and then updates IBM MQ resources within the global unit of work by issuing MQPUT, MQPUT1, and MQGET calls within syncpoint control. The application can then commit the global unit of work by calling ENDTRANSACTION, or back it out by calling ABORTTRANSACTION.

An application that is using TMF transactions can only actively work on one transaction at any one time, however using RESUMETRANSACTION allows an application to switch from one active transaction to another, or to being associated with no TMF transaction, without completing or aborting the previously active transaction. Any calls to MQPUT, MQPUT1, or MQGET are made under the currently active TMF transaction, if present, or a local unit of work, if not present. Therefore, care must be taken within the application to ensure that these calls are being made within the correct unit of work.

Within a global unit of work, as well as updating IBM MQ resources, an application can update Enscribe files, SQL/MP databases, or SQL/MX databases.

Using global units of work

NSS Client

A global unit of work is implemented as a TMF transaction. An application starts a global unit of work by calling BEGINTRANSACTION, and either commits the unit of work by calling ENDTRANSACTION or backs out the unit of work by calling ABORTTRANSACTION. An application can use other TMF API calls as well.

An application can inherit a TMF transaction from another application. For example, an application (the first application) can perform work within the transaction before replying and passing the transaction back to a second application for further processing. Both the first and the second applications can therefore participate in the same global unit of work that involves updates to IBM MQ queues and updates to files and databases. The ability to pass a TMF transaction between applications means that several IBM MQ applications can perform messaging operations within the same global unit of work.

An application can manage and control multiple active TMF transactions at the same time. The transactions can be started by the application itself, or inherited from other applications, or both. This means that an application can participate in multiple global units of work at the same time.

The maximum number of concurrent active TMF transactions per process is 1000, which is an architectural limit. If an application is managing multiple TMF transactions, only one transaction can be current at any point in time. Alternatively, none of the transactions can be current. The application can use TMF API calls such as RESUMETRANSACTION, ACTIVATERECEIVETRANSID, and TMF_SET_TX_ID to move the state of being current from one transaction to another, or to designate that no transaction is current. The application uses this level of control to determine whether a messaging operation is performed within a local unit of work, a global unit of work, or outside of syncpoint control:

- If an application calls MQPUT, MQPUT1, or MQGET within syncpoint control when no TMF transaction is current, IBM MQ processes the call within a local unit of work.
- If an application calls MQPUT, MQPUT1, or MQGET within syncpoint control when the application has a current TMF transaction, IBM MQ processes the call within the global unit of work that is implemented by the current TMF transaction.
- If an application calls MQPUT, MQPUT1, or MQGET outside of syncpoint control, IBM MQ processes the call outside of syncpoint control, irrespective of whether the application has a current TMF transaction at the time of the call.

IBM MQ never changes the state of an application's TMF transaction during an MQI call, except when a software or hardware failure occurs during processing and IBM MQ or the operating system determines that the transaction must be backed out to preserve data integrity. Every MQI call restores the transaction state of the application just before returning control to the application.

Avoiding long running transactions

NSS Client

Avoid designing applications in which TMF transactions remain active for more than a few tens of seconds. Long running transactions can cause the circular audit trail of TMF to fill up. Because TMF is a critical system-wide resource, TMF protects itself by backing out application transactions that are active for too long.

Suppose that the processing within an application is driven by getting messages from a queue, and that the application gets a message from the queue and processes the message within a unit of work. Typically, an application calls MQGET with the wait option and within syncpoint control to get a message from the queue.

If the application is using a global unit of work instead, the specified wait interval on the MQGET call must be short to avoid a long running transaction. This means that the application might need to issue the MQGET call more than once before it retrieves a message.

Planning your IBM MQ environment on z/OS

z/OS

When planning your IBM MQ environment, you must consider the resource requirements for data sets, page sets, Db2, Coupling Facilities, and the need for logging, and backup facilities. Use this topic to plan the environment where IBM MQ runs.

Before you plan your IBM MQ architecture, familiarize yourself with the basic IBM MQ for z/OS concepts, see the topics in IBM MQ for z/OS concepts.

Related concepts:

"Planning an IBM MQ architecture" on page 1
When planning your IBM MQ environment, consider the support that IBM MQ provides for single and multiple queue manager architectures, and for point-to-point and publish/subscribe messaging styles. Also plan your resource requirements, and your use of logging and backup facilities.

Related information:

IBM MQ Technical overview

Configuring z/OS

Administering IBM MQ for z/OS

Planning your storage and performance requirements on z/OS

z/OS

You must set realistic and achievable storage, and performance goals for your IBM MQ system. Use this topic help you understand the factors which affect storage, and performance.

This topic contains information about the storage and performance requirements for IBM MQ for z/OS. It contains the following sections:

- z/OS performance options for IBM MQ
- Determining z/OS workload management importance and velocity goals
- "Library storage" on page 140
- "System LX usage" on page 140
- "Address space storage" on page 140
- "Data storage" on page 144

See, “Where to find more information about storage and performance requirements” on page 145 for more information.

z/OS performance options for IBM MQ

With workload management, you define performance goals and assign a business importance to each goal. You define the goals for work in business terms, and the system decides how much resource, such as processor and storage, should be given to the work to meet its goal. Workload management controls the dispatching priority based on the goals you supply. Workload management raises or lowers the priority as needed to meet the specified goal. Thus, you need not fine-tune the exact priorities of every piece of work in the system and can focus instead on business objectives.

The three kinds of goals are:

Response time
How quickly you want the work to be processed

Execution velocity
How fast the work should be run when ready, without being delayed for processor, storage, I/O access, and queue delay

Discretionary
A category for low priority work for which there are no performance goals

Response time goals are appropriate for end-user applications. For example, CICS® users might set workload goals as response time goals. For IBM MQ address spaces, velocity goals are more appropriate. A small amount of the work done in the queue manager is counted toward this velocity goal but this work is critical for performance. Most of the work done by the queue manager counts toward the performance goal of the end-user application. Most of the work done by the channel initiator address space counts toward its own velocity goal. The receiving and sending of IBM MQ messages, which the channel initiator accomplishes, is typically important for the performance of business applications using them.

Determining z/OS workload management importance and velocity goals

For full information about workload management and defining goals through the service definition, see *z/OS MVS Planning: Workload Management*.

This section suggests how to set the z/OS workload management importance and velocity goals relative to other important work in your system.

Use the following service classes:

The default SYSSTC service class
- VTAM and TCP/IP address spaces
- IRLM address space (IRLMPROC)

Note: The VTAM, TCP/IP, and IRLM address spaces must have a higher dispatching priority than all the DBMS address spaces, their attached address spaces, and their subordinate address spaces. Do not allow workload management to reduce the priority of VTAM, TCP/IP, or IRLM to (or below) that of the other DBMS address spaces

A high velocity goal and importance of 1 for a service class with a name that you define, such as PRODREGN, for the following:
- IBM MQ queue manager and channel initiator address spaces
- Db2 (all address spaces, except for the Db2-established stored procedures address space)
- CICS (all region types)

- IMS™ (all region types except BMPs)

  A high velocity goal is good for ensuring that startups and restarts are performed as quickly as possible for all these address spaces.

The velocity goals for CICS and IMS regions are only important during startup or restart. After transactions begin running, workload management ignores the CICS or IMS velocity goals and assigns priorities based on the response time goals of the transactions that are running in the regions. These transaction goals should reflect the relative priority of the business applications they implement. They might typically have an importance value of 2. Any batch applications using MQ should similarly have velocity goals and importance reflecting the relative priority of the business applications they implement. Typically the importance and velocity goals will be less than those for PRODREGN.

Library storage

You must allocate storage for the product libraries. The exact figures depend on your configuration, but an estimate of the space required by the distribution libraries is 80 MB. The target libraries require about 72 MB. Additionally, you require space for the SMP/E libraries.

The target libraries used by IBM MQ for z/OS use PDS or PDSE formats. Ensure that any PDSE target libraries are not shared outside a sysplex. For more information about the required libraries and their sizes and the required format, see the *Program Directory for IBM MQ for z/OS*, which can be downloaded from the from the IBM Publications Center (see IBM MQ Version 9.0 PDF documentation).

System LX usage

Each defined IBM MQ subsystem reserves one system linkage index (LX) at IPL time, and a number of non-system linkage indexes when the queue manager is started. The system linkage index is reused when the queue manager is stopped and restarted. Similarly, distributed queuing reserves one non-system linkage index. In the unlikely event of your z/OS system having inadequate system LXs defined, you might need to take these reserved system LXs into account.

Address space storage

z/OS

Use this topic for basic guidance on address space requirements for the IBM MQ components.

Storage requirements can be divided into the following categories:
- Common storage
- Queue manager private region storage usage
- Channel initiator storage usage

For more details see, Suggested regions sizes.

With 31-bit address space, a virtual line marks the 16-megabyte address, and 31-bit addressable storage is often known as “above the (16MB) line”. With 64-bit address space there is a second virtual line called the bar that marks the 2-gigabyte address. The bar separates storage below the 2-gigabyte address, called “below the bar”, from storage above the 2-gigabyte address, called “above the bar”. Storage below the bar uses 31-bit addressability, storage above the bar uses 64-bit addressability.

You can specify the limit of 31-bit storage by using the REGION parameter on the JCL, and the limit of above the bar storage by using the MEMLIMIT parameter. These specified values can be overridden by MVS exits.

Common storage

Each IBM MQ for z/OS subsystem has the following approximate storage requirements:

- CSA 4 KB
- ECSA 800 KB, plus the size of the trace table that is specified in the TRACTBL parameter of the CSQ6SYSP system parameter macro. For more information, see Using CSQ6SYSP.

In addition, each concurrent IBM MQ logical connection requires about 5 KB of ECSA. When a task ends, other IBM MQ tasks can reuse this storage. IBM MQ does not release the storage until the queue manager is shut down, so you can calculate the maximum amount of ECSA required by multiplying the maximum number of concurrent logical connections by 5 KB. Concurrent logical connections are the number of:

- Tasks (TCBs) in Batch, TSO, z/OS UNIX and Linux System Services, IMS, and Db2 SPAS regions that are connected to IBM MQ, but not disconnected.
- CICS transactions that have issued an IBM MQ request, but have not terminated
- JMS Connections, Sessions, TopicSessions or QueueSessions that have been created (for bindings connection), but not yet destroyed or garbage collected.
- Active IBM MQ channels.

You can set a limit to the common storage, used by logical connections to the queue manager, with the ACELIM configuration parameter. The ACELIM control is primarily of interest to sites where Db2 stored procedures cause operations on IBM MQ queues.

When driven from a stored procedure, each IBM MQ operation can result in a new logical connection to the queue manage. Large Db2 units of work, for example due to table load, can result in an excessive demand for common storage.

ACELIM is intended to limit common storage use and to protect the z/OS system. Using ACELIM causes IBM MQ failures when the limit is exceeded. See the ACELIM section in Using CSQ6SYSP for more information.

Use SupportPac MP1B to format the SMF 115 subtype 3 records produced by STATISTICS CLASS(2) trace.

The amount of storage currently in the subpool controlled by the ACELIM value is indicated in the output, on the line titled *ACE/PEB*. SupportPac MP1B indicates the number of bytes in use.

Increase the normal value by a sufficient margin to provide space for growth and workload spikes. Divide the new value by 1024 to yield a maximum storage size in KB for use in the ACELIM configuration.

The channel initiator typically requires ECSA usage of up to 160 KB.

Queue manager private region storage usage

IBM MQ for z/OS can use storage above the 2 GB bar for some internal control blocks. You can have buffer pools in this storage, which gives you the potential to configure much larger buffer pools if sufficient storage is available. Typically buffer pools are the major internal control blocks that use storage above the 2 GB bar.

Each buffer pool size is determined at queue manager initialization time, and storage is allocated for the buffer pool when a page set that is using that buffer pool is connected. A new parameter LOCATION (ABOVE|BELOW) is used to specify where the buffers are allocated. You can use the ALTER BUFFPOOL command to dynamically change the size of buffer pools.

To use above the bar (64 Bit) storage, you can specify a value for MEMLIMIT parameter (for example MEMLIMIT=3G) on the **`EXEC PGM=CSQYASCP`** parameter in the queue manager JCL. Your installation might have a default value set.

You should specify a MEMLIMIT and specify a sensible storage size rather than MEMLIMIT=NOLIMIT to prevent potential problems. If you specify NOLIMIT or a very large value, then an ALTER BUFFPOOL command with a large size, can use up all of the available z/OS virtual storage, which will lead to paging in your system.

Start with a MEMLIMIT=3G and increase this size when you need to increase the size of your buffer pools.

Specify MEMLIMIT= 2 GB plus the size of the buffer pools above the bar, rounded up to the nearest GB. For example, for 2 buffer pools configured with LOCATION ABOVE, buffer pool 1 has 10,000 buffers, buffer pool 2 has 50,000 buffers. Memory usage above the bar equals 60,000 (total number of buffers) * 4096 = 245,760,000 bytes = 199.3 MB, rounded up to 200MB as a multiple of 4MB. All buffer pools regardless of LOCATION will make use of 64 bit storage for control structures. As the number of buffer pools and number of buffers in those pools increase this can become significant. A good rule of thumb is that each buffer requires an additional 200 bytes of 64 bit storage. For a configuration with 10 buffer pools each with 20,000 buffers that would require: 200 * 10 * 20,000 = 40,000,000 equivalent to 40 MB. You can specify 3 GB for the MEMLIMIT size, which will allow scope for growth (40MB + 200MB + 2 GB which rounds up to 3 GB).

For some configurations there can be significant performance benefits to using buffer pools that have their buffers permanently backed by real storage. You can achieve this by specifying the FIXED4KB value for the PAGECLAS attribute of the buffer pool. However, you should only do this if there is sufficient real storage available on the LPAR, otherwise other address spaces might be affected. For information about when you should use the FIXED4KB value for PAGECLAS, see IBM MQ Support Pac MP16: IBM MQ for z/OS - Capacity planning & tuning

To minimize paging, consider real storage in addition to the virtual storage that is used by the queue manager and the channel initiator.

Before you use storage above the bar, you should discuss with your MVS systems programmer to ensure that there is sufficient auxiliary storage for peak time usage, and sufficient real storage requirements to prevent paging.

Note:
The size of memory dump data sets might have to be increased to handle the increased virtual storage.

Making the buffer pools so large that there is MVS paging might adversely affect performance. You might consider using a smaller buffer pool that does not page, with IBM MQ moving the message to and from the page set.

You can monitor the address space storage usage from the CSQY220I message that indicates the amount of private region storage in use above and below the 2 GB bar, and the remaining amount.

Channel initiator storage usage

There are two areas of channel initiator storage usage that you must consider:
- Private region
- Accounting and statistics

Private region storage usage

You should specify REGION=0M for the CHINIT to allow it to use the maximum below the bar storage. The storage available to the channel initiator limits the number of concurrent connections the CHINIT can have.

Every channel uses approximately 170 KB of extended private region in the channel initiator address space. Storage is increased by message size if messages larger than 32 KB are transmitted. This increased storage is freed when:

- A sending or client channel requires less than half the current buffer size for 10 consecutive messages.
- A heartbeat is sent or received.

The storage is freed for reuse within the Language Environment, however, is not seen as free by the z/OS virtual storage manager. This means that the upper limit for the number of channels is dependent on message size and arrival patterns, and on limitations of individual user systems on extended private region size. The upper limit on the number of channels is likely to be approximately 9000 on many systems because the extended region size is unlikely to exceed 1.6 GB. The use of message sizes larger than 32 KB reduces the maximum number of channels in the system. For example, if messages that are 100 MB long are transmitted, and an extended region size of 1.6 GB is assumed, the maximum number of channels is 15.

The channel initiator trace is written to a dataspace. The size of the database storage, is controlled by the **`TRAXTBL`** parameter. See ALTER QMGR.

Accounting and statistics storage usage

You should allow the channel initiator access to a minimum of 256 MB of virtual storage, and you can do this by specifying `MEMLIMIT=256M`.

If you do not set the MEMLIMIT parameter in the channel initiator JCL, you can set the amount of virtual storage above the bar using the MEMLIMIT parameter in the `SMFPRMxx` member of SYS1.PARMLIB, or from the IEFUSI exit.

If you set the MEMLIMIT to restrict the above bar storage below the required level, the channel initiator issues the CSQX124E message and class 4 accounting and statistics trace will not be available.

Suggested region sizes

The following table shows suggested values for region sizes.

Table 17. Suggested definitions for JCL region sizes

Definition setting	System
Queue manager	REGION=0M, MEMLIMIT=3G
Channel initiator	REGION=0M

Managing the MEMLIMIT and REGION size

Other mechanisms, for example the **`MEMLIMIT`** parameter in the SMFPRMxx member of SYS1.PARMLIB or the IEFUSI exit might be used at your installation to provide a default amount of virtual storage above the bar for z/OS address spaces. See the z/OSMVS Programming: Extended Addressability Guide for full details about limiting storage above the bar.

Data storage

z/OS

Use this topic when planning your data storage requirements for log data sets, Db2 storage, coupling facility storage, and page data sets.

Work with your storage administrator to determine where to put the queue manager data sets. For example, your storage administrator may give you specific DASD volumes, or SMS storage classes, data classes, and management classes for the different data set types.

- Log data sets must be on DASD. These logs can have high I/O activity with a small response time and do not need to be backed up.
- Archive logs can be on DASD or tape. After they have been created, they might never be read again except in an abnormal situation, such as recovering a page set from a backup. They should have a long retention date.
- Page sets might have low to medium activity and should be backed up regularly. On a high use system, they should be backed up twice a day.
- BSDS data sets should be backed up daily; they do not have high I/O activity.

All data sets are similar to those used by Db2, and similar maintenance procedures can be used for IBM MQ.

See the following sections for details of how to plan your data storage:

- **Logs and archive storage**

  "How long do I need to keep archive logs" on page 169 describes how to determine how much storage your active log and archive data sets require, depending on the volume of messages that your IBM MQ system handles and how often the active logs are offloaded to your archive data sets.
- **Db2 storage**

  "Db2 storage" on page 162 describes how to determine how much storage Db2 requires for the IBM MQ data.
- **coupling facility storage**

  "Defining coupling facility resources" on page 153 describes how to determine how large to make your coupling facility structures.
- **Page set and message storage**

  "Planning your page sets and buffer pools" on page 145 describes how to determine how much storage your page data sets require, depending on the sizes of the messages that your applications exchange, on the numbers of these messages, and on the rate at which they are created or exchanged.

Where to find more information about storage and performance requirements

z/OS

Use this topic as a reference to find more information about storage and performance requirements.

You can find more information from the following sources:

Table 18. Where to find more information about storage requirements

Topic	Where to look
System parameters	Using CSQ6SYSP and Customizing your queue managers
Storage required to install IBM MQ	Program Directory for IBM MQ for z/OS, which can be downloaded from the IBM Publications Center (see IBM MQ Version 9.0 PDF documentation).
IEALIMIT and IEFUSI exits	*MVS Installation Exits*, available from the zSeries website z/OS Internet Library.
Latest information	IBM MQ SupportPac Web site Business Integration - WebSphere MQ SupportPacs.
Workload management and defining goals through the service definition	*z/OS MVS Planning: Workload Management*

Planning your page sets and buffer pools

z/OS

Information to help you with planning the initial number, and sizes of your page data sets, and buffer pools.

This topic contains the following sections:
- "Plan your page sets"
  - Page set usage
  - Number of page sets
  - Size of page sets
- "Calculate the size of your page sets" on page 146
  - Page set zero
  - Page set 01 - 99
  - Calculating the storage requirement for messages
- "Enabling dynamic page set expansion" on page 149
- "Defining your buffer pools" on page 150

Plan your page sets

Page set usage

For short-lived messages, few pages are normally used on the page set and there is little or no I/O to the data sets except at startup, during a checkpoint, or at shutdown.

For long-lived messages, those pages containing messages are normally written out to disk. This operation is performed by the queue manager in order to reduce restart time.

Separate short-lived messages from long-lived messages by placing them on different page sets and in different buffer pools.

Number of page sets

Using several large page sets can make the role of the IBM MQ administrator easier because it means that you need fewer page sets, making the mapping of queues to page sets simpler.

Using multiple, smaller page sets has a number of advantages. For example, they take less time to back up, and I/O can be carried out in parallel during backup and restart. However, consider that this adds a significant performance cost to the role of the IBM MQ administrator, who is required to map each queue to one of a much greater number of page sets.

Define at least five page sets, as follows:
- A page set reserved for object definitions (page set zero)
- A page set for system-related messages
- A page set for performance-critical long-lived messages
- A page set for performance-critical short-lived messages
- A page set for all other messages

"Defining your buffer pools" on page 150 explains the performance advantages of distributing your messages on page sets in this way.

Size of page sets

Define sufficient space in your page sets for the expected peak message capacity. Consider for any unexpected peak capacity, such as when a build-up of messages develops because a queue server program is not running. You can be do this by allocating the page set with secondary extents or, alternatively, by enabling dynamic page set expansion. For more information, see "Enabling dynamic page set expansion" on page 149.

When planning page set sizes, consider all messages that might be generated, including non-application message data. For example, trigger messages, event messages and any report messages that your application has requested.

The size of the page set determines the time taken to recover a page set when restoring from a backup, because a large page set takes longer to restore.

Note: Recovery of a page set also depends on the time the queue manager takes to process the log records written since the backup was taken; this time period is determined by the backup frequency. For more information, see "Planning for backup and recovery" on page 176.

Note: Page sets larger than 4 GB require the use of SMS extended addressability.

Calculate the size of your page sets

For queue manager object definitions (for example, queues and processes), it is simple to calculate the storage requirement because these objects are of fixed size and are permanent. For messages however, the calculation is more complex for the following reasons:
- Messages vary in size.
- Messages are transitory.
- Space occupied by messages that have been retrieved is reclaimed periodically by an asynchronous process.

Large page sets of greater than 4 GB that provide extra capacity for messages if the network stops, can be created if required. It is not possible to modify the existing page sets. Instead, new page sets with extended addressability and extended format attributes, must be created. The new page sets must be the same physical size as the old ones, and the old page sets must then be copied to the new ones. If backward migration is required, page set zero must not be changed. If page sets less than 4 GB are adequate, no action is needed.

Page set zero

For page set zero, the storage required is:

```
   (maximum number of local queue definitions x 1010)
      (excluding shared queues)
+  (maximum number of model queue definitions x 746)
+  (maximum number of alias queue definitions x 338)
+  (maximum number of remote queue definitions x 434)
+  (maximum number of permanent dynamic queue definitions x 1010)
+  (maximum number of process definitions x 674)
+  (maximum number of namelist definitions x 12320)
+  (maximum number of message channel definitions x 2026)
+  (maximum number of client-connection channel definitions x 5170)
+  (maximum number of server-connection channel definitions x 2026)
+  (maximum number of storage class definitions x 266)
+  (maximum number of authentication information definitions x 1010)
+  (maximum number of administrative topic definitions x 15000)
+  (total length of topic strings defined in administrative topic definitions)
```

Divide this value by 4096 to determine the number of records to specify in the cluster for the page set data set.

You do not need to allow for objects that are stored in the shared repository, but you must allow for objects that are stored or copied to page set zero (objects with a disposition of GROUP or QMGR).

The total number of objects that you can create is limited by the capacity of page set zero. The number of local queues that you can define is limited to 524 287.

Page sets 01 - 99

For page sets 01 - 99, the storage required for each page set is determined by the number and size of the messages stored on that page set. (Messages on shared queues are not stored on page sets.)

Divide this value by 4096 to determine the number of records to specify in the cluster for the page set data set.

Calculating the storage requirement for messages

This section describes how messages are stored on pages. Understanding this can help you calculate how much page set storage you must define for your messages. To calculate the approximate space required for all messages on a page set you must consider maximum queue depth of all the queues that map to the page set and the average size of messages on those queues.

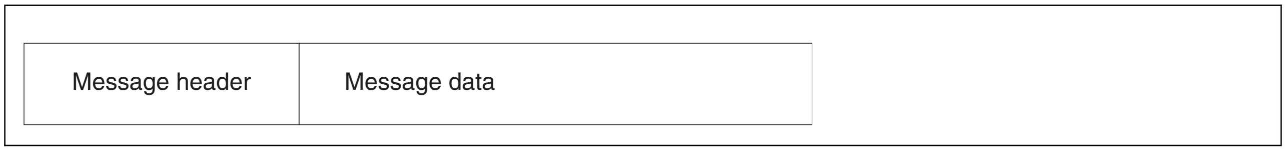
Note: The sizes of the structures and control information given in this section are liable to change between major releases. For details specific to your release of IBM MQ, refer to SupportPac MP16 - WebSphere MQ for z/OS Capacity planning & tuning and MP1E / MP1F / MP1G - WebSphere MQ for z/OS Vx.x.x Performance report

You must allow for the possibility that message "gets" might be delayed for reasons outside the control of IBM MQ (for example, because of a problem with your communications protocol). In this case, the "put" rate of messages might far exceed the "get" rate. This can lead to a large increase in the number of messages stored in the page sets and a consequent increase in the storage size demanded.

Each page in the page set is 4096 bytes long. Allowing for fixed header information, each page has 4057 bytes of space available for storing messages.

When calculating the space required for each message, the first thing you must consider is whether the message fits on one page (a short message) or whether it needs to be split over two or more pages (a long message). When messages are split in this way, you must allow for additional control information in your space calculations.

For the purposes of space calculation, a message can be represented as the following:



The message header section contains the message descriptor and other control information, the size of which varies depending on the size of the message. The message data section contains all the actual message data, and any other headers (for example, the transmission header or the IMS bridge header).

A minimum of two pages are required for page set control information which, is typically less than 1% of the total space required for messages.

Short messages

A short message is defined as a message that fits on one page.

From IBM WebSphere MQ Version 7.0.1, small messages are stored one on each page.

Long messages

If the size of the message data is greater than 3596 bytes, but not greater than 4 MB, the message is classed as a long message. When presented with a long message, IBM MQ stores the message on a series of pages, and stores control information that points to these pages in the same way that it would store a short message. This is shown in Figure 55:

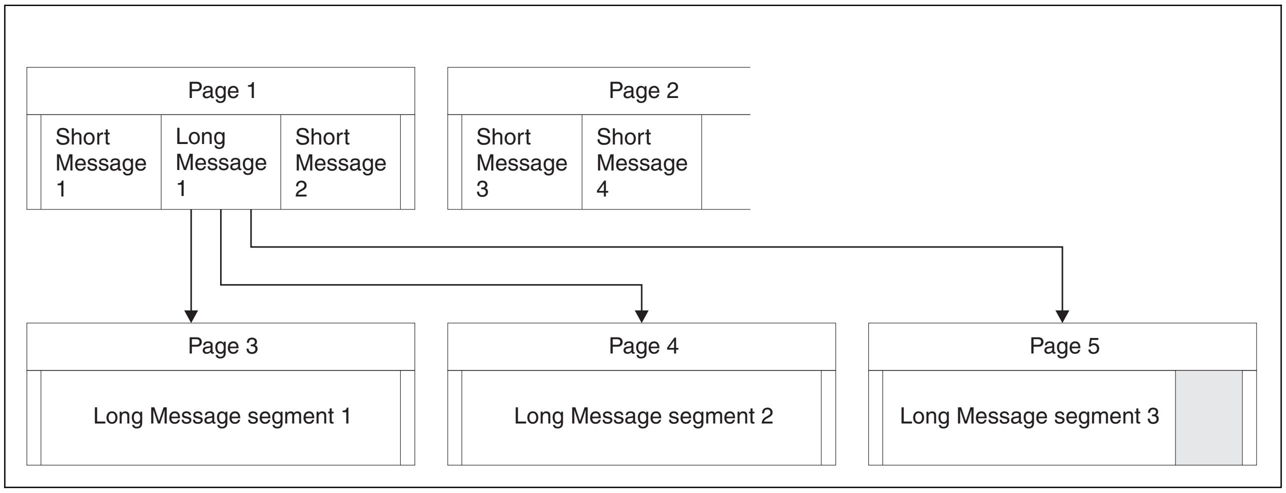


Figure 55. How IBM MQ stores long messages on page sets

Very long messages

Very long messages are messages with a size greater than 4 MB. These are stored so that each 4 MB uses 1037 pages. Any remainder is stored in the same way as a long message, as described above.

Enabling dynamic page set expansion

z/OS

Page sets can be extended dynamically while the queue manager is running. A page set can have 119 extents, and can be spread over multiple disk volumes.

Each time a page set expands, a new data set extent is used. The queue manager continues to expand a page set when required, until the maximum number of extents has been reached, or until no more storage is available for allocation on eligible volumes.

Once page set expansion fails for one of the reasons above, the queue manager marks the page set for no further expansion attempts. This marking can be reset by altering the page set to EXPAND(SYSTEM).

Page set expansion takes place asynchronously to all other page set activity, when 90% of the existing space in the page set is allocated.

The page set expansion process formats the newly allocated extent and makes it available for use by the queue manager. However, none of the space is available for use, until the entire extent has been formatted. This means that expansion by a large extent is likely to take some time, and putting applications might 'block' if they fill the remaining 10% of the page set before the expansion has completed.

Sample thlqual.SCSQPROC(CSQ4PAGE) shows how to define the secondary extents.

To control the size of new extents, you use one of the following options of the EXPAND keyword of the DEFINE PSID and ALTER PSID commands:
- USER
- SYSTEM
- NONE

USER

Uses the secondary extent size specified when the page set was allocated. If a value was not specified, or if a value of zero was specified, dynamic page set expansion cannot occur.

Page set expansion occurs when the space in the page is 90% used, and is performed asynchronously with other page set activity.

This may lead to expansion by more than a single extent at a time.

Consider the following example: you allocate a page set with a primary extent of 100000 pages and a secondary extent of 5000 pages. A message is put that requires 9999 pages. If the pageset is already using 85,000 pages, writing the message crosses the 90% full boundary (90,000 pages). At this point, a further secondary extent is allocated to the primary extent of 100,000 pages, taking the page set size to 105,000 pages. The remaining 4999 pages of the message continue to be written. When the used page space reaches 94,500 pages, which is 90% of the updated page set size of 105,000 pages, another 5000 page extent is allocated, taking the page set size to 110,000 pages. At the end of the MQPUT, the pageset has expanded twice, and 94,500 pages are used. None of the ages in the second page set expansion have been used, although they were allocated.

At restart, if a previously used page set has been replaced with a data set that is smaller, it is expanded until it reaches the size of the previously used data set. Only one extent is required to reach this size.

SYSTEM

Ignores the secondary extent size that was specified when the page set was defined. Instead, the queue manager sets a value that is approximately 10% of the current page set size. The value is rounded up to the nearest cylinder of DASD.

If a value was not specified, or if a value of zero was specified, dynamic page set expansion can still occur. The queue manager sets a value that is approximately 10% of the current page set size. The new value is rounded up depending on the characteristics of the DASD.

Page set expansion occurs when the space in the page set is approximately 90% used, and is performed asynchronously with other page set activity.

At restart, if a previously used page set has been replaced with a data set that is smaller, it is expanded until it reaches the size of the previously used data set.

NONE

No further page set expansion is to take place.

Related information:

ALTER PSID

DEFINE PSID

DISPLAY USAGE

Defining your buffer pools

Use this topic to help plan the number of buffer pools you should define, and their settings.

This topic is divided into the following sections:

1. "Decide on the number of buffer pools to define"
2. "Decide on the initial characteristics of each buffer pool" on page 151
3. "Monitor the performance of buffer pools under expected load" on page 152
4. "Adjust buffer pool characteristics" on page 152

Decide on the number of buffer pools to define

You should define four buffer pools initially:

Buffer pool 0

Use for object definitions (in page set zero) and performance critical, system related message queues, such as the SYSTEM.CHANNEL.SYNCQ queue and the SYSTEM.CLUSTER.* queues.

Use the remaining three buffer pools for user messages.

Buffer pool 1

Use for important long-lived messages.

Long-lived messages are those that remain in the system for longer than two checkpoints, at which time they are written out to the page set. If you have many long-lived messages, this buffer pool should be relatively small, so that page set I/O is evenly distributed (older messages are written out to DASD each time the buffer pool becomes 85% full).

If the buffer pool is too large, and the buffer pool never gets to 85% full, page set I/O is delayed until checkpoint processing. This might affect response times throughout the system.

If you expect few long-lived messages only, define this buffer pool so that it is sufficiently large to hold all these messages.

Buffer pool 2

Use for performance-critical, short-lived messages.

There is normally a high degree of buffer reuse, using few buffers. However, you should make this buffer pool large to allow for unexpected message accumulation, for example, when a server application fails.

Buffer pool 3

Use for all other (typically, performance noncritical) messages.

Queues such as the dead-letter queue, SYSTEM.COMMAND.* queues and SYSTEM.ADMIN.* queues can also be mapped to buffer pool 3.

Where virtual storage constraints exist, and buffer pools need to be smaller, buffer pool 3 is the first candidate for size reduction.

You might need to define additional buffer pools in the following circumstances:
- If a particular queue is known to require isolation, perhaps because it exhibits different behavior at various times.
  - Such a queue might either require the best performance possible under the varying circumstances, or need to be isolated so that it does not adversely affect the other queues in a buffer pool.
  - Each such queue can be isolated into its own buffer pool and pageset.
- You want to isolate different sets of queues from each other for class-of-service reasons.
  - Each set of queues might then require one, or both, of the two types of buffer pools 1 or 2, as described in Table 20 on page 152, necessitating creation of several buffer pools of a specific type.

CD In IBM MQ Version 9.0 for z/OS, the maximum number of buffer pools that you can define depends on the OPMODE parameter. If IBM MQ Version 8.0 new functions are enabled, a maximum of 100 buffer pools can be defined. Otherwise, a maximum of 16 buffer pools can be defined.

Decide on the initial characteristics of each buffer pool

Having decided on the number of buffer pools that you require, you now need to decide the initial characteristics of those buffer pools. The available characteristics are affected by OPMODE, and are summarized in Table 19.

Table 19. Buffer pool characteristics by OPMODE setting

Parameter name/ OPMODE setting	**Maximum number of buffers in a single buffer pool**	**Maximum number of buffer pools**	**Maximum total number of buffers in queue manager**	**Buffer pool location**	**Buffer pool page class**
DEFINE or ALTER BUFFPOOL parameter	BUFFERS			LOCATION (ABOVE/ BELOW)	PAGECLAS (4KB/FIXED4KB)
CD IBM MQ Version 8.0 new functions not enabled	500 000	16	Limited by available space below the bar. Likely to be less than 262 144	BELOW bar	Pageable (4KB)
CD IBM MQ Version 8.0 functions enabled	999 999 999	100	99 999 999 900 (If MEMLIMIT /IEFUSI exit allows)	ABOVE or BELOW bar	Pageable (4KB) or page-fixed (FIXED4KB)

If you are using the four buffer pools described in "Decide on the number of buffer pools to define" on page 150, then Table 20 on page 152 gives two sets of values for the size of the buffer pools.

The first set is suitable for a test system, the other for a production system or a system that will become a production system eventually. Regardless of the OPMODE setting, the values assume that the buffer pools will be located below the bar, and that the buffers will be pageable.

Table 20. Suggested definitions for buffer pool settings

Definition setting	Test system	Production system
BUFFPOOL 0	1 050 buffers	50 000 buffers
BUFFPOOL 1	1 050 buffers	20 000 buffers
BUFFPOOL 2	1 050 buffers	50 000 buffers
BUFFPOOL 3	1 050 buffers	20 000 buffers

If you need more than the four suggested buffer pools, select the buffer pool (1 or 2) that most accurately describes the expected behavior of the queues in the buffer pool, and size it using the information in Table 20.

CD It might be that you will need to reduce the size of some of the other buffer pools, or reconsider the number of buffer pools, especially if you have not enabled IBM MQ Version 8.0 new functions with OPMODE.

Monitor the performance of buffer pools under expected load

You can monitor the usage of buffer pools by analyzing buffer pool performance statistics. In particular, you should ensure that the buffer pools are large enough so that the values of QPSTSOS, QPSTSTLA, and QPSTDMC remain at zero.

For further information, see Buffer manager data records.

Adjust buffer pool characteristics

Use the following points to adjust the buffer pool settings from "Decide on the initial characteristics of each buffer pool" on page 151, if required.

Use the performance statistics from "Monitor the performance of buffer pools under expected load" as guidance.

Note: CD These points all assume that IBM MQ Version 8.0 new functions have been enabled with OPMODE, with the exception of point 2.

1. If you are migrating from an earlier version of IBM MQ, only change your existing settings if you have more real storage available.
2. In general, bigger buffer pools are better for performance, and buffer pools can be much bigger if they are above the bar.

   However, at all times you should have sufficient real storage available so that the buffer pools are resident in real storage. It is better to have smaller buffer pools that do not result in paging, than big ones that do.

   Additionally, there is no point having a buffer pool that is bigger than the total size of the pagesets that use it, although you should take into account pageset expansion if it is likely to occur.
3. Aim for one buffer pool per pageset, as this provides better application isolation.
4. If you have sufficient real storage, such that your buffer pools will never be paged out by the operating system, consider using page-fixed buffers in your buffer pool.

   This is particularly important if the buffer pool is likely to undergo much I/O, as it saves the CPU cost associated with page-fixing the buffers before the I/O, and page-unfixing them afterwards.
5. There are several benefits to locating buffer pools above the bar even if they are small enough to fit below the bar. These are:
   - 31 bit virtual storage constraint relief - for example more space for common storage.

  - If the size of a buffer pool needs to be increased unexpectedly while it is being heavily used, there is less impact and risk to the queue manager, and its workload, by adding more buffers to a buffer pool that is already above the bar, than moving the buffer pool to above the bar and then adding more buffers.
6. Tune buffer pool zero and the buffer pool for short-lived messages (buffer pool 2) so that the 15% free threshold is never exceeded (that is, QPSTCBSL divided by QPSTNBUF is always greater than 15%). If more than 15% of buffers remain free, I/O to the page sets using these buffer pools can be largely avoided during normal operation, although messages older than two checkpoints are written to page sets.

   Attention: The optimum value for these parameters is dependent on the characteristics of the individual system. The values given are intended only as a guideline and might not be appropriate for your system.
7. SYSTEM.* queues which get very deep, for example SYSTEM.CHANNEL.SYNCQ, might benefit from being placed in their own buffer pool, if sufficient storage is available.

IBM MQ SupportPac MP16 - WebSphere MQ for z/OS Capacity planning & tuning provides further information about tuning buffer pools.

Planning your coupling facility and offload storage environment

z/OS

Use this topic when planning the initial sizes, and formats of your coupling facility (CF) structures, and shared message data set (SMDS) environment or Db2 environment.

This section contains information about the following topics:
- "Defining coupling facility resources"
  - Deciding your offload storage mechanism
  - Planning your structures
  - Planning the size of your structures
  - Mapping shared queues to structures
- "Planning your shared message data set (SMDS) environment" on page 158
- "Planning your Db2 environment" on page 162

Defining coupling facility resources

If you intend to use shared queues, you must define the coupling facility structures that IBM MQ will use in your CFRM policy. To do this you must first update your CFRM policy with information about the structures, and then activate the policy.

Your installation probably has an existing CFRM policy that describes the Coupling Facilities available. The IXCMIAPU z/OS utility is used to modify the contents of the policy based on textual statements you provide. The utility is described in the *MVS Setting up a Sysplex* manual. You must add statements to the policy that define the names of the new structures, the Coupling Facilities that they are defined in, and what size the structures are.

The CFRM policy also determines whether IBM MQ structures are duplexed and how they are reallocated in failure scenarios. Shared queue recovery contains recommendations for configuring CFRM for System Managed Rebuild processing.

Deciding your offload storage environment

The message data for shared queues can be offloaded from the coupling facility and stored in either a Db2 table or in an IBM MQ managed data set called a *shared message data set* (SMDS). Messages which are too large to store in the coupling facility (that is, larger than 63 KB) must always be offloaded, and smaller messages may optionally be offloaded to reduce coupling facility space usage.

For more information, see Specifying offload options for shared messages.

Planning your structures

A queue-sharing group requires a minimum of two structures to be defined. The first structure, known as the administrative structure, is used to coordinate IBM MQ internal activity across the queue-sharing group. No user data is held in this structure. It has a fixed name of *qsg-name* CSQ_ADMIN (where *qsg-name* is the name of your queue-sharing group). Subsequent structures are used to hold the messages on IBM MQ shared queues. Each structure can hold up to 512 shared queues.

Using multiple structures

A queue-sharing group can connect to up to 64 coupling facility structures. One of these structures must be the administration structure, one of these structures might be the SYSAPPL structure. So you can use up to 63 (62 with SYSAPPL) structures for IBM MQ data. You might choose to use multiple structures for any of the following reasons:

- You have some queues that are likely to hold a large number of messages and so require all the resources of an entire coupling facility.
- You have a requirement for a large number of shared queues, so they must be split across multiple structures because each structure can contain only 512 queues.
- RMF reports on the usage characteristic of a structure suggest that you should distribute the queues it contains across a number of Coupling Facilities.
- You want some queue data to held in a physically different coupling facility from other queue data for data isolation reasons.
- Recovery of persistent shared messages is performed using structure level attributes and commands, for example BACKUP CFSTRUCT. To simplify backup and recovery, you could assign queues that hold nonpersistent messages to different structures from those structures that hold persistent messages.

When choosing which Coupling Facilities to allocate the structures in, consider the following points:

- Your data isolation requirements.
- The volatility of the coupling facility (that is, its ability to preserve data through a power outage).
- Failure independence between the accessing systems and the coupling facility, or between Coupling Facilities.
- The level of coupling facility Control Code (CFCC) installed on the coupling facility (IBM MQ requires Level 9 or higher).

Planning the size of your structures

The administrative structure (*qsg-name* CSQ_ADMIN) must be large enough to contain 1000 list entries for each queue manager in the queue-sharing group. When a queue manager starts, the structure is checked to see if it is large enough for the number of queue managers currently *defined* to the queue-sharing group. Queue managers are considered as being defined to the queue-sharing group if they have been added by the CSQ5PQSG utility. You can check which queue managers are defined to the group with the MQSC DISPLAY GROUP command.

Table 21 on page 155 shows the minimum required size for the administrative structure for various numbers of queue managers defined in the queue-sharing group. These sizes were established for a CFCC level 14 coupling facility structure; for higher levels of CFCC, they probably need to be larger.

Table 21. Minimum administrative structure sizes

Number of queue managers defined in queue-sharing group	Required storage
1	6144 KB
2	6912 KB
3	7976 KB
4	8704 KB
5	9728 KB
6	10496 KB
7	11520 KB
8	12288 KB
9	13056 KB
10	14080 KB
11	14848 KB
12	15616 KB
13	16640 KB
14	17408 KB
15	18176 KB
16	19200 KB
17	19968 KB
18	20736 KB
19	21760 KB
20	22528 KB
21	23296 KB
22	24320 KB
23	25088 KB
24	25856 KB
25	27136 KB
26	27904 KB
27	28672 KB
28	29696 KB
29	30464 KB
30	31232 KB
31	32256 KB

When you add a queue manager to an existing queue-sharing group, the storage requirement might have increased beyond the size recommended in Table 21. If so, use the following procedure to estimate the required storage for the CSQ_ADMIN structure: Issue MQSC command `/pf DISPLAY CFSTATUS(*)`, where /cpf is for an existing member of the queue-sharing group, and extract the ENTSMAX information for the CSQ_ADMIN structure. If this number is less than 1000 times the total number of queue managers you want to define in the queue-sharing group (as reported by the DISPLAY GROUP command), increase the structure size.

The size of the structures required to hold IBM MQ messages depends on the likely number and size of the messages to be held on a structure concurrently, together with an estimate of the likely number of concurrent units of work.

The graph in Figure 56 shows how large you should make your CF structures to hold the messages on your shared queues. To calculate the allocation size you need to know

- The average size of messages on your queues
- The total number of messages likely to be stored in the structure

Find the number of messages along the horizontal axis. (Ticks are at multiples of 2, 5, and 8.) Select the curve that corresponds to your message size and determine the required value from the vertical axis. For example, for 200 000 messages of length 1 KB gives a value in the range 256 through 512MB.

Table 22 on page 157 provides the same information in tabular form.

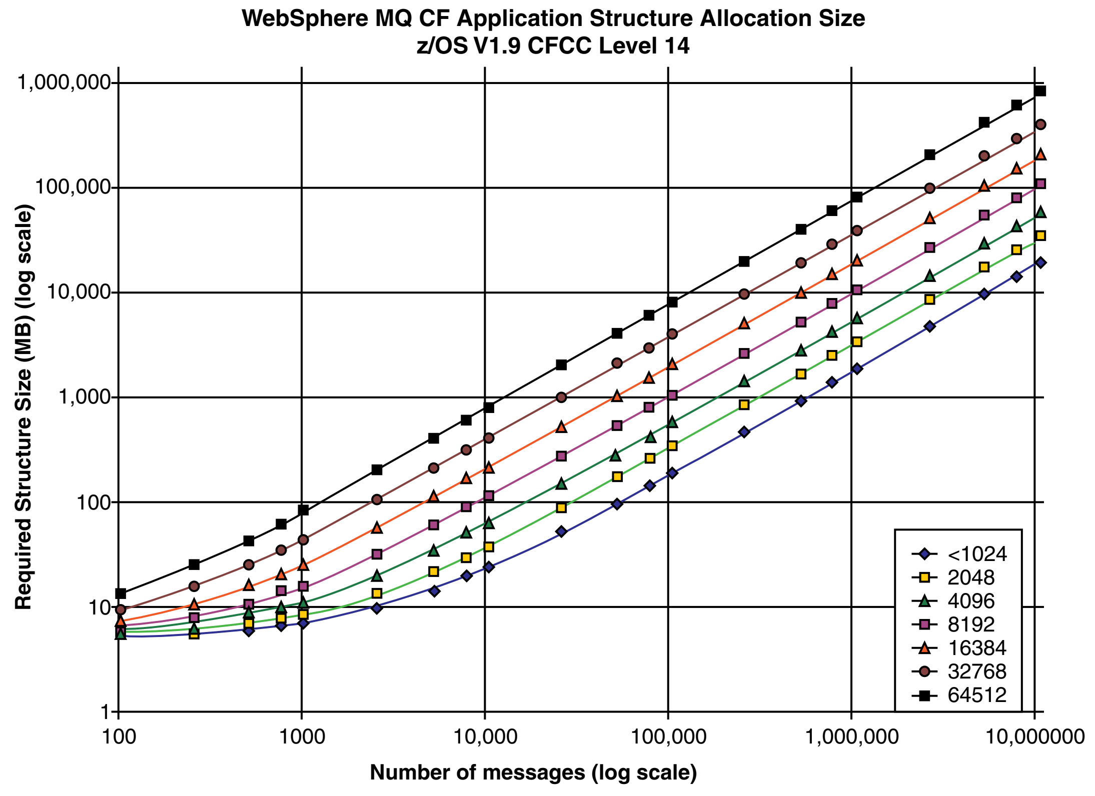


Figure 56. Calculating the size of a coupling facility structure

Use this table to help calculate how large to make your coupling facility structures:

Table 22. Calculating the size of a coupling facility structure

Number of messages	1 KB	2 KB	4 KB	8 KB	16 KB	32 KB	63 KB
100	6	6	7	7	8	10	14
1000	8	9	12	17	27	48	88
10000	25	38	64	115	218	423	821
100000	199	327	584	1097	2124	4177	8156

Your CFRM policy should include the following statements:

`INITSIZE` is the size in KB that XES allocates to the structure when the first connector connects to it. `SIZE` is the maximum size that the structure can attain. `FULLTHRESHOLD` sets the percentage value of the threshold at which XES issues message IXC585E to indicate that the structure is getting full. A best practice is to ensure that `INITSIZE` and `SIZE` are within a factor of 2.

For example, with the figures determined previously, you might include the following statements:

```
STRUCTURE NAME(structure-name)
INITSIZE(value from graph in KB, that is, multiplied by 1024)
SIZE(something larger)
FULLTHRESHOLD(85)
```

```
STRUCTURE NAME(QSG1APPLICATION1)
INITSIZE(272144) /* 256 MB */
SIZE(524288) /* 512 MB */
FULLTHRESHOLD(85)
```

If the structure use reaches the threshold where warning messages are issued, intervention is required. You might use IBM MQ to inhibit `MQPUT` operations to some of the queues in the structure to prevent applications from writing more messages, start more applications to get messages from the queues, or quiesce some of the applications that are putting messages to the queue.

Alternatively, you can use XES facilities to alter the structure size in place. The following z/OS command:

```
SETXCF START,ALTER,STRNAME= structure-name,SIZE= newsize
```

alters the size of the structure to *`newsize`*, where *`newsize`* is a value that is less than the value of SIZE specified on the CFRM policy for the structure, but greater than the current coupling facility size.

You can monitor the use of a coupling facility structure with the MQSC DISPLAY GROUP command.

If no action is taken and a queue structure fills up, an MQRC_STORAGE_MEDIUM_FULL return code is returned to the application. If the administration structure becomes full, the exact symptoms depend on which processes experience the error, but they might include the following problems:
- No responses to commands.
- Queue manager failure as a result of problems during commit processing.

Starting in V7.0.1 certain system queues are provided with CFSTRUCT attributes which specify an application structure CSQSYSAPPL prefixed with the queue-sharing group name. The CSQSYSAPPL structure is an application structure for system queues. For details of creating the coupling facility structures see Task 10: Set up the coupling facility.

With the default definitions, the SYSTEM.QSG.CHANNEL.SYNCQ and SYSTEM.QSG.UR.RESOLUTION.QUEUE use this structure. Table 3 demonstrates an example of how to estimate the message data sizes for the default queues.

Table 23. Table showing CSQSYSAPPL usage against sizing.

qsg-name **CSQSYSAPPL usage**	**sizing**
SYSTEM.QSG.CHANNEL.SYNCQ	2 messages of 500 bytes per active instance of a shared channel
SYSTEM.QSG.UR.RESOLUTION.QUEUE	1000 messages of 2 KB

The suggested initial structure definition values are as follows:

```
STRUCTURE NAME(qsgname CSQSYSAPPL)
INITSIZE(20480)          /* 20 MB */
SIZE(30720)            /* 30 MB */
FULLTHRESHOLD(85)
```

These values can be adjusted depending on your use of shared channels and group units of recovery.

Mapping shared queues to structures

The CFSTRUCT attribute of the queue definition is used to map the queue to a structure.

IBM MQ adds the name of the queue-sharing group to the beginning of the CFSTRUCT attribute. For a structure defined in the CFRM policy with name *qsg-name* SHAREDQ01, the definition of a queue that uses this structure is:

```
DEFINE QLOCAL( myqueue ) QSGDISP(SHARED) CFSTRUCT(SHAREDQ01)
```

Planning your shared message data set (SMDS) environment

z/OS

If you are using queue-sharing groups with SMDS offloading, IBM MQ needs to connect to a group of shared message data sets. Use this topic to help understand the data set requirements, and configuration required to store IBM MQ message data.

A *shared message data set* (described by the keyword SMDS) is a data set used by a queue manager to store offloaded message data for shared messages stored in a coupling facility structure.

When this form of data offloading is enabled, the **CFSTRUCT** requires an associated group of shared message data sets, one data set for each queue manager in the queue-sharing group. The group of shared message data sets is defined to IBM MQ using the **DSGROUP** parameter on the **CFSTRUCT** definition. Additional parameters can be used to supply further optional information, such as the number of buffers to use and expansion attributes for the data sets.

Each queue manager can write to the data set which it owns, to store shared message data for messages written via that queue manager, and can read all of the data sets in the group

A list describing the status and attributes for each data set associated with the structure is maintained internally as part of the **CFSTRUCT** definition, so each queue manager can check the definition to find out which data sets are currently available.

This data set information can be displayed using the **DISPLAY CFSTATUS TYPE(SMDS)** command to display current status and availability, and the **DISPLAY SMDS** command to display the parameter settings for the data sets associated with a specified **CFSTRUCT**.

Individual shared message dta sets are effectively identified by the combination of the owning queue manager name (usually specified using the **SMDS** keyword) and the **CFSTRUCT** structure name.

This section describes the following topics:

- The DSGROUP parameter
- The DSBLOCK parameter
- Shared message data set characteristics
- Shared message data set space management
- Access to shared message data sets
- Creating a shared message data set
- Shared message data set performance and capacity considerations
- Activating a shared message data set

See DEFINE CFSTRUCT for details of these parameters.

For information on managing your shared message data sets, see Managing shared message data sets for further details.

The DSGROUP parameter

The **DSGROUP** parameter on the **CFSTRUCT** definition identifies the group of data sets in which large messages for that structure are to be stored. Additional parameters may be used to specify the logical block size to be used for space allocation purposes and values for the buffer pool size and automatic data set expansion options.

The **DSGROUP** parameter must be set up before offloading to data sets can be enabled.
- If a new **CFSTRUCT** is being defined at **CFLEVEL(5)** and the option **OFFLOAD(SMDS)** is specified or assumed, then the **DSGROUP** parameter must be specified on the same command.
- If an existing **CFSTRUCT** is being altered to increase the **CFLEVEL** to **CFLEVEL(5)** and the option **OFFLOAD(SMDS)** is specified or assumed, then the **DSGROUP** parameter must be specified on the same command if it is not already set.

The DSBLOCK parameter

Space within each data set is allocated to queues as logical blocks of a fixed size (usually 256 KB) specified using the **DSBLOCK** parameter on the **CFSTRUCT** definition, then allocated to individual messages as ranges of pages of 4 KB (corresponding to the physical block size and control interval size) within each logical block. The logical block size also determines the maximum amount of message data that can be read or written in a single I/O operation, which is the same as the buffer size for the SMDS buffer pool.

A larger value of the **DSBLOCK** parameter can improve performance for very large messages by reducing the number of separate I/O operations. However, a smaller value decreases the amount of buffer storage required for each active request. The default value for the **DSBLOCK** parameter is 256 KB, which provides a reasonable balance between these requirements, so specifying this parameter might not normally be necessary.

Shared message data set characteristics

A shared message data set is defined as a VSAM linear data set (LDS). Each offloaded message is stored in one or more blocks in the data set. The stored data is addressed directly by information in the coupling facility entries, like an extended form of virtual storage. There is no separate index or similar control information stored in the data set itself.

The direct addressing scheme means that for messages which fit into one block, only a single I/O operation is needed to read or write the block. When a message spans more than one block, the I/O operations for each block can be fully overlapped to minimize elapsed time, provided that sufficient buffers are available.

The shared message data set also contains a small amount of general control information, consisting of a header in the first page, which includes recovery and restart status information, and a space map checkpoint area which is used to save the free block space map at queue manager normal termination.

Shared message data set space management

As background information for capacity, performance and operational considerations, it might be useful to understand the concepts of how space in shared message data sets is managed by the queue managers.

Free space in each shared message data set is tracked by its owning queue manager using a space map which indicates the number of pages in use within each logical block. The space map is maintained in main storage while the data set is open and saved in the data set when it is closed normally. (In recovery situations the space map is automatically rebuilt by scanning the messages in the coupling facility structure to find out which data set pages are currently in use).

When a shared message with offloaded message data is being written, the queue manager allocates a range of pages for each message block. If there is a partly used current logical block for the specified queue, the queue manager allocates space starting at the next free page in that block, otherwise it allocates a new logical block. If the whole message does not fit within the current logical block, the queue manager splits the message data at the end of the logical block and allocates a new logical block for the next message block. This is repeated until space has been allocated for the whole message. Any unused space in the last logical block is saved as the new current logical block for the queue. When the data set is closed normally, any unused pages in current logical blocks are returned to the space map before it is saved.

When a shared message with offloaded message data has been read and is ready to be deleted, the queue manager processes the delete request by transferring the coupling facility entry for the message to a clean-up list monitored by the owning queue manager (which may be the same queue manager). When entries arrive on this list, the owning queue manager reads and deletes the entries and returns the freed ranges of pages to the space map. When all used pages in a logical block have been freed the block becomes available for reuse.

Access to shared message data sets

Each shared message data set must be on shared direct access storage which is accessible to all queue managers in the queue-sharing group.

During normal running, each queue manager opens its own shared message data set for read/write access, and opens any active shared message data sets for other queue managers for read-only access, so it can read messages stored by those queue managers. This means that each queue manager userid requires at least UPDATE access to its own shared message data set and READ access to all other shared message data sets for the structure.

If it is necessary to recover shared message data sets using **RECOVER CFSTRUCT**, the recovery process can be executed from any queue manager in the queue-sharing group. A queue manager which may be used to perform recovery processing requires UPDATE access to all data sets that it may need to recover

Creating a shared message data set

Each shared message data set should normally be created before the corresponding **CFSTRUCT** definition is created or altered to enable the use of this form of message offloading, as the **CFSTRUCT** definition changes will normally take effect immediately, and the data set will be required as soon as a queue manager attempts to access a shared queue which has been assigned to that structure. A sample job to allocate and pre-format a shared message data set is provided in SCSQPROC(CSQ4SMDS). The job must be customized and run to allocate a shared message data set for each queue manager which uses a CFSTRUCT with OFFLOAD(SMDS).

If the queue manager finds that offload support has been enabled and tries to open its shared message data set but it has not yet been created, the shared message data set will be flagged as unavailable. The queue manager will then be unable to store any large messages until the data set has been created and the queue manager has been notified to try again, for example using the **START SMDSCONN** command.

A shared message data set is created as a VSAM linear data set using an Access Method Services **DEFINE CLUSTER** command. The definition must specify **SHAREOPTIONS(2 3)** to allow one queue manager to open it for write access and any number of queue managers to read it at the same time. The default control interval size of 4 KB must be used. If the data set may need to expand beyond 4 GB, it must be defined using an SMS data class which has the VSAM extended addressability attribute.

Each shared message data set can either be empty or pre-formatted to binary zeros (using **CSQJUFMT** or a similar utility such as the sample job SCSQPROC(CSQ4SMDS)), before its initial use. If it is empty or only partly formatted when it is opened, the queue manager automatically formats the remaining space to binary zeros.

Shared message data set performance and capacity considerations

Each shared message data set is used to store offloaded data for shared messages written to the associated **CFSTRUCT** by the owning queue manager, from regions within the same system. The stored data for each message includes a descriptor (currently about 350 bytes), the message headers and the message body. Each offloaded message is stored in one or more pages (physical blocks of size 4 KB) in the data set.

The data set space required for a given number of offloaded messages can therefore be estimated by rounding up the overall message size (including the descriptor) to the next multiple of 4 KB and then multiplying by the number of messages.

As for a page set, when a shared message data set is almost full, it can optionally be expanded automatically. The default behavior for this automatic expansion can be set using the **DSEXPAND** parameter on the **CFSTRUCT** definition. This setting can be overridden for each queue manager using the **DSEXPAND** parameter on the **ALTER SMDS** command. Automatic expansion is triggered when the data set reaches 90% full and more space is required. If expansion is allowed but an expansion attempt is rejected by VSAM because no secondary space allocation was specified when the data set was defined, expansion is retried using a secondary allocation of 20% of the current size of the data set.

Provided that the shared message data set is defined with the extended addressability attribute, the maximum size is only limited by VSAM considerations to a maximum of 16 TB or 59 volumes. This is significantly larger than the 64 GB maximum size of a local page set.

Activating a shared message data set

When a queue manager has successfully connected to an application coupling facility structure, it checks whether that structure definition specifies offloading using an associated **DSGROUP** parameter. If so, the queue manager allocates and opens its own shared message data set for write access, then it opens for read access any existing shared message data sets owned by other queue managers.

When a shared message data set is opened for the first time (before it has been recorded as active within the queue-sharing group), the first page will not yet contain a valid header. The queue manager fills in header information to identify the queue-sharing group, the structure name and the owning queue manager.

After the header has been completed, the queue manager registers the new shared message data set as active and broadcasts an event to notify any other active queue managers about the new data set.

Every time a queue manager opens a shared message data set it validates the header information to ensure that the correct data set is still being used and that it has not been damaged.

Planning your Db2 environment

z/OS

If you are using queue-sharing groups, IBM MQ needs to attach to a Db2 subsystem that is a member of a data sharing group. Use this topic to help understand the Db2 requirements used to hold IBM MQ data.

IBM MQ needs to know the name of the data sharing group that it is to connect to, and the name of a Db2 subsystem (or Db2 group) to connect to, to reach this data sharing group. These names are specified in the QSGDATA parameter of the CSQ6SYSP system parameter macro (described in Using CSQ6SYSP).

Within the data sharing group, shared Db2 tables are used to hold:
- Configuration information for the queue-sharing group.
- Properties of IBM MQ shared and group objects.
- Optionally, data relating to offloaded IBM MQ messages.

V 9.0.4 IBM MQ provides sample jobs for defining the Db2 tablespaces, tables, and indexes. Two sets of sample jobs are provided:
- One for compatibility with earlier versions of IBM MQ
- One for use with Db2 V12 and later, which exploits Universal Table Spaces (UTS)

Since Db2 V11, various types of table space, used by earlier versions of IBM MQ, have been flagged as deprecated. Where possible, when setting up a new data sharing group you are encouraged to choose the set of jobs which exploit Db2 UTS.

V 9.0.4 Use of preexisting table space types remains supported. There is no non-disruptive migration path from these non-preferred table spaces to UTS. Tables must be unloaded, redefined in UTS, and reloaded, necessitating an IBM MQ queue-sharing group outage.

By default Db2 uses the user ID of the person running the jobs as the owner of the Db2 resources. If this user ID is deleted then the resources associated with it are deleted, and so the table is deleted. Consider using a group ID to own the tables, rather than an individual user ID. You can do this by adding `GROUP=groupname` onto the JOB card, and specifying `SET CURRENT SQLID='groupname'` before any SQL statements.

IBM MQ uses the RRS Attach facility of Db2. This means that you can specify the name of a Db2 group that you want to connect to. The advantage of connecting to a Db2 group attach name (rather than a specific Db2 subsystem), is that IBM MQ can connect (or reconnect) to any available Db2 subsystem on the z/OS image that is a member of that group. There must be a Db2 subsystem that is a member of the data sharing group active on each z/OS image where you are going to run a queue-sharing IBM MQ subsystem, and RRS must be active.

Db2 storage

For most installations, the amount of Db2 storage required is about 20 or 30 cylinders on a 3390 device. However, if you want to calculate your storage requirement, the following table gives some information to help you determine how much storage Db2 requires for the IBM MQ data. The table describes the length of each Db2 row, and when each row is added to or deleted from the relevant Db2 table. Use this information together with the information about calculating the space requirements for the Db2 tables and their indexes in the *Db2 for z/OS Installation Guide*.

Table 24. Planning your Db2 storage requirements

Db2 table name	Length of row	A row is added when:	A row is deleted when:
CSQ.ADMIN_B_QSG	252 bytes	A queue-sharing group is added to the table with the ADD QSG function of the CSQ5PQSG utility.	A queue-sharing group is removed from the table with the REMOVE QSG function of the CSQ5PQSG utility. (All rows relating to this queue-sharing group are deleted automatically from all the other Db2 tables when the queue-sharing group record is deleted.)
CSQ.ADMIN_B_QMGR	Up to 3828 bytes	A queue manager is added to the table with the ADD QMGR function of the CSQ5PQSG utility.	A queue manager is removed from the table with the REMOVE QMGR function of the CSQ5PQSG utility.
CSQ.ADMIN_B_STRUCTURE	1454 bytes	The first local queue definition, specifying the QSGDISP(SHARED) attribute, that names a previously unknown structure within the queue-sharing group is defined.	The last local queue definition, specifying the QSGDISP(SHARED) attribute, that names a structure within the queue-sharing group is deleted.
CSQ.ADMIN_B_SCST	342 bytes	A shared channel is started.	A shared channel becomes inactive.
CSQ.ADMIN_B_SSKT	254 bytes	A shared channel that has the NPMSPEED(NORMAL) attribute is started.	A shared channel that has the NPMSPEED(NORMAL) attribute becomes inactive.
CSQ.ADMIN_B_STRBACKUP	514 bytes	A new row is added to the CSQ.ADMIN_B_STRUCTURE table. Each entry is a dummy entry until the BACKUP CFSTRUCT command is run, which overwrites the dummy entries.	A row is deleted from the CSQ.ADMIN_B_STRUCTURE table.
CSQ.OBJ_B_AUTHINFO	3400 bytes	An authentication information object with QSGDISP(GROUP) is defined.	An authentication information object with QSGDISP(GROUP) is deleted.
CSQ.OBJ_B_QUEUE	Up to 3707 bytes	• A queue with the QSGDISP(GROUP) attribute is defined. • A queue with the QSGDISP(SHARED) attribute is defined. • A model queue with the DEFTYPE(SHAREDYN) attribute is opened.	• A queue with the QSGDISP(GROUP) attribute is deleted. • A queue with the QSGDISP(SHARED) attribute is deleted. • A dynamic queue with the DEFTYPE(SHAREDYN) attribute is closed with the DELETE option.
CSQ.OBJ_B_NAMELIST	Up to 15127 bytes	A namelist with the QSGDISP(GROUP) attribute is defined.	A namelist with the QSGDISP(GROUP) attribute is deleted.
CSQ.OBJ_B_CHANNEL	Up to 14127 bytes	A channel with the QSGDISP(GROUP) attribute is defined.	A channel with the QSGDISP(GROUP) attribute is deleted.
CSQ.OBJ_B_STGCLASS	Up to 2865 bytes	A storage class with the QSGDISP(GROUP) attribute is defined.	A storage class with the QSGDISP(GROUP) attribute class is deleted.

Table 24. Planning your Db2 storage requirements (continued)

Db2 table name	Length of row	A row is added when:	A row is deleted when:
CSQ.OBJ_B_PROCESS	Up to 3347 bytes	A process with the QSGDISP(GROUP) attribute is defined.	A process with the QSGDISP(GROUP) attribute is deleted.
CSQ.OBJ_B_TOPIC	Up to 14520 bytes	A topic object with QSGDISP(GROUP) attribute is defined.	A topic object with QSGDISP(GROUP) attribute is deleted.
CSQ.EXTEND_B_QMGR	Less than 430 bytes	A queue manager is added to the table with the ADD QMGR function of the CSQ5PQSG utility.	A queue manager is removed from the table with the REMOVE QMGR function of the CSQ5PQSG utility.
CSQ.ADMIN_B_MESSAGES	87 bytes	For large message PUT (1 per BLOB).	For large message GET (1 per BLOB).
CSQ.ADMIN_MSGS_BAUX1 CSQ.ADMIN_MSGS_BAUX2 CSQ.ADMIN_MSGS_BAUX3 CSQ.ADMIN_MSGS_BAUX4		These 4 tables contain message payload for large messages added into one of these 4 tables for each BLOB of the message. BLOBS are up to 511 KB in length, so if the message size is > 711 KB, there will be multiple BLOBs for this message.	

The use of large numbers of shared queue messages of size greater than 63 KB can have significant performance implications on your IBM MQ system. For more information, see SupportPac MP16, Capacity Planning and Tuning for IBM MQ for z/OS, at: Business Integration - IBM MQ SupportPacs.

Planning your logging environment

z/OS

Use this topic to plan the number, size and placement of the logs, and log archives used by IBM MQ.

Logs are used to:
- Write recovery information about persistent messages
- Record information about units of work using persistent messages
- Record information about changes to objects, such as define queue
- Backup CF structures

and for other internal information.

The IBM MQ logging environment is established using the system parameter macros to specify options, such as: whether to have single or dual active logs, what media to use for the archive log volumes, and how many log buffers to have.

These macros are described in Task 14: Create the bootstrap and log data sets and Task 17: Tailor your system parameter module.

Note: If you are using queue-sharing groups, ensure that you define the bootstrap and log data sets with SHAREOPTIONS(2 3).

This section contains information about the following topics:

Log data set definitions

z/OS

Use this topic to decide on the most appropriate configuration for your log data sets.

This topic contains information to help you answer the following questions:
- Should your installation use single or dual logging?
- How many active log data sets do you need?
- “How large should the active logs be?” on page 166
- Active log placement

Should your installation use single or dual logging?

In general you should use dual logging for production, to minimize the risk of losing data. If you want your test system to reflect production, both should use dual logging, otherwise your test systems can use single logging.

With single logging data is written to one set of log data sets. With dual logging data is written to two sets of log data sets, so in the event of a problem with one log data set, such as the data set being accidentally deleted, the equivalent data set in the other set of logs can be used to recover the data.

With dual logging you require twice as much DASD as with single logging.

If you are using dual logging, then also use dual BSDSs and dual archiving to ensure adequate provision for data recovery.

Dual active logging adds a small performance cost.

Attention: Always use dual logging and dual BSDSs rather than dual writing to DASD (mirroring). If a mirrored data set is accidentally deleted, both copies are lost.

If you use persistent messages, single logging can increase maximum capacity by 10-30% and can also improve response times.

Single logging uses 2 - 31 active log data sets, whereas dual logging uses 4 - 62 active log data sets to provide the same number of active logs. Thus single logging reduces the amount of data logged, which might be important if your installation is I/O constrained.

How many active log data sets do you need?

The number of logs depends on the activities of your queue manager. For a test system with low throughput, three active log data sets might be suitable. For a high throughput production system you might want the maximum number of logs available, so, if there is a problem with offloading logs you have more time to resolve the problems.

You must have at least three active log data sets, but it is preferable to define more. For example, if the time taken to fill a log is likely to approach the time taken to archive a log during peak load, define more logs.

You should also define more logs to offset possible delays in log archiving. If you use archive logs on tape, allow for the time required to mount the tape.

Consider having enough active log space to keep a day's worth of data, in case the system is unable to archive because of lack of DASD or because it cannot write to tape.

It is possible to dynamically define new active log data sets as a way of minimizing the effect of archive delays or problems. New data sets can be brought online rapidly, using the DEFINE LOG command to avoid queue manager 'stall' due to lack of space in the active log.

If you want to define more than 31 active log data sets, you must configure your logging environment to use a version 2 format BSDS. Once a version 2 format BSDS is in use, up to 310 active log data sets can be defined for each log copy ring. See “Planning to increase the maximum addressable log range” on page 171 for information on how you convert to a version 2 format BSDS.

You can tell whether your queue manager is using a version 2 or higher BSDS, either by running the print log map utility (CSQJU004), or from the CSQJ034I message issued during queue manager initialization. An end of log RBA range of `FFFFFFFFFFFFFFFF`, in the CSQJ034I message, indicates that a version 2, or higher, format BSDS is in use.

When a queue manager is using a version 2, or higher, format BSDS it is possible to use the DEFINE LOG command to dynamically add more than 31 active log data sets to a log copy ring.

How large should the active logs be?

From IBM MQ Version 8.0, the maximum supported active log size, when archiving to disk, is 4 GB. In previous releases of the product, the maximum supported active log size when archiving to disk is 3 GB.

When archiving to tape the maximum active log size is 4 GB.

You should create active logs of at least 1 GB in size for production and test systems.

Important: You need to be careful when allocating data sets, because IDCAMS rounds up the size you allocate.

To allocate a 3 GB log specify one of the following options:
- Cylinders(4369)
- Megabytes(3071)
- TRACKS(65535)
- RECORD(786420)

Any one of these allocates 2.99995 GB.

To allocate a 4GB log specify one of the following options:
- Cylinders(5825)
- Megabytes(4095)
- TRACKS(87375)
- RECORD(1048500)

Any one of these allocates 3.9997 GB.

When using striped data sets, where the data set is spread across multiple volumes, the specified size value is allocated on each DASD volume used for striping. So, if you want to use 4 GB logs and four volumes for striping, you should specify:
- CYLinders(1456)
- Megabytes(1023)

Setting these attributes allocates 4*1456 = 5824 Cylinders or 4 * 1023 = 4092 Megabytes.

Note: Striping is supported when using extended format data sets. This is usually set by the storage manager.

Active log placement

For performance reasons you should consider striping your active log data sets. The I/O is spread across multiple volumes and reduces the I/O response times, leading to higher throughput. See the preceding text for information about allocating the size of the active logs when using striping.

You should review the I/O statistics using reports from RMF or a similar product., Perform the review of these statistics monthly (or more frequently) for the IBM MQ data sets, to ensure there are no delays due to the location of the data sets.

In some situations, there can be much IBM MQ page set I/O, and this can impact the IBM MQ log performance if they are located on the same DASD.

If you use dual logging, ensure that each set of active and archive logs is kept apart. For example, allocate them on separate DASD subsystems, or on different devices.

This reduces the risk of them both being lost if one of the volumes is corrupted or destroyed. If both copies of the log are lost, the probability of data loss is high.

When you create a new active log data, set you should preformat it using CSQJUFMT. If the log is not preformatted, the queue manager formats the log the first time it is used, which impacts the performance.

With older DASD with large spinning disks, you had to be careful which volumes were used to get the best performance.

With modern DASD, where data is spread over many PC sized disks, you do not need to worry so much about which volumes are used.

Your storage manager should be checking the enterprise DASD to review and resolve any performance problems. For availability, you might want to use one set of logs on one DASD subsystem, and the dual logs on a different DASD subsystem.

Planning your log archive storage

z/OS

Use this topic to understand the different ways of maintaining your archive log data sets.

You can place archive log data sets on standard-label tapes, or DASD, and you can manage them by data facility hierarchical storage manager (DFHSM). Each z/OS logical record in an archive log data set is a VSAM control interval from the active log data set. The block size is a multiple of 4 KB.

Archive log data sets are dynamically allocated, with names chosen by IBM MQ. The data set name prefix, block size, unit name, and DASD sizes needed for such allocations are specified in the system parameter module. You can also choose, at installation time, to have IBM MQ add a date and time to the archive log data set name.

It is not possible to specify with IBM MQ, specific volumes for new archive logs, but you can use Storage Management routines to manage this. If allocation errors occur, offloading is postponed until the next time offloading is triggered.

If you specify dual archive logs at installation time, each log control interval retrieved from the active log is written to two archive log data sets. The log records that are contained in the pair of archive log data sets are identical, but the end-of-volume points are not synchronized for multivolume data sets.

Should your archive logs reside on tape or DASD?

When deciding whether to use tape or DASD for your archive logs, there are a number of factors that you should consider:

- Review your operating procedures before deciding about tape or disk. For example, if you choose to archive to tape, there must be enough tape drive when they are required. After a disaster, all subsystems might want tape drives and you might not have as many free tape drives as you expect.
- During recovery, archive logs on tape are available as soon as the tape is mounted. If DASD archives have been used, and the data sets migrated to tape using hierarchical storage manager (HSM), there is a delay while HSM recalls each data set to disk. You can recall the data sets before the archive log is used. However, it is not always possible to predict the correct order in which they are required.
- When using archive logs on DASD, if many logs are required (which might be the case when recovering a page set after restoring from a backup) you might require a significant quantity of DASD to hold all the archive logs.
- In a low-usage system or test system, it might be more convenient to have archive logs on DASD to eliminate the need for tape mounts.
- Both issuing a RECOVER CFSTRUCT command, and backing out a persistent unit of work, result in the log being read backwards. Tape drives with hardware compression perform badly on operations that read backwards. Plan sufficient log data on DASD to avoid reading backwards from tape.

Archiving to DASD offers faster recoverability but is more expensive than archiving to tape. If you use dual logging, you can specify that the primary copy of the archive log go to DASD and the secondary copy go to tape. This increases recovery speed without using as much DASD, and you can use the tape as a backup.

Archiving to tape

If you choose to archive to a tape device, IBM MQ can extend to a maximum of 20 volumes.

If you are considering changing the size of the active log data set so that the set fits on one tape volume, note that a copy of the BSDS is placed on the same tape volume as the copy of the active log data set. Adjust the size of the active log data set downward to offset the space required for the BSDS on the tape volume.

If you use dual archive logs on tape, it is typical for one copy to be held locally, and the other copy to be held off-site for use in disaster recovery.

Archiving to DASD volumes

IBM MQ requires that you catalog all archive log data sets allocated on non-tape devices (DASD). If you choose to archive to DASD, the CATALOG parameter of the CSQ6ARVP macro must be YES. If this parameter is NO, and you decide to place archive log data sets on DASD, you receive message CSQJ072E each time an archive log data set is allocated, although IBM MQ still catalogs the data set.

If the archive log data set is held on DASD, the archive log data sets can extend to another volume; multivolume is supported.

If you choose to use DASD, make sure that the primary space allocation (both quantity and block size) is large enough to contain either the data coming from the active log data set, or that from the corresponding BSDS, whichever is the larger of the two.

This minimizes the possibility of unwanted z/OS `X'B37'` or `X'E37'` abend codes during the offload process. The primary space allocation is set with the PRIQTY (primary quantity) parameter of the CSQ6ARVP macro.

From IBM MQ Version 8.0, archive log data sets can exist on large or extended-format sequential data sets. SMS ACS routines can now use DSNTYPE(LARGE) or DSNTYPE(EXT). These were not supported before Version 8.0.

IBM MQ supports allocation of archive logs as extended format data sets. When extended format is used, the maximum archive log size is increased from 65535 tracks to the maximum active log size of 4GB. Archive logs are eligible for allocation in the extended addressing space (EAS) of extended address volumes (EAV).

Where the required hardware and software levels are available, allocating archive logs to a data class defined with COMPACTION using zEDC might reduce the disk storage required to hold archive logs. For more information, see IBM MQ for z/OS: Reducing storage occupancy with IBM zEnterprise Data Compression (zEDC).

Refer to Using the zEnterprise Data Compression (zEDC) enhancements for details on hardware and software levels, as well as example RACF profile changes.

The z/OS data set encryption feature can be applied to archive logs for queue managers running at IBM MQ Version 8.0 or later. These archive logs must be allocated through Automatic Class Selection (ACS) routines to a data class defined with EXTENDED attributes, and a data set key label that ensures the data is AES encrypted.

Using SMS with archive log data sets

If you have MVS™/DFP storage management subsystem (DFSMS) installed, you can write an Automatic Class Selection (ACS) user-exit filter for your archive log data sets, which helps you convert them for the SMS environment.

Such a filter, for example, can route your output to a DASD data set, which DFSMS can manage. You must exercise caution if you use an ACS filter in this manner. Because SMS requires DASD data sets to be cataloged, you must make sure the CATALOG DATA field of the CSQ6ARVP macro contains YES. If it does not, message CSQJ072E is returned; however, the data set is still cataloged by IBM MQ.

For more information about ACS filters, see DFP Storage Administration Reference.

How long do I need to keep archive logs

z/OS

Use the information in this section to help you plan your backup strategy.

You specify how long archive logs are kept in days , using the ARCRETN parameter in USING CSQ6ARVP or the SET SYSTEM command. After this period the data sets can be deleted by z/OS.

You can manually delete archive log data sets when they are no longer needed.

- The queue manager might need the archive logs for recovery.

  The queue manager can only keep the most recent 1000 archives in the BSDS, When the archive logs are not in the BSDS they cannot be used for recovery, and are only of use for audit, analysis, or replay type purposes.
- You might want to keep the archive logs so that you can extract information from the logs. For example, extracting messages from the log, and reviewing which user ID put or got the message.

The BSDS contains information on logs and other recovery information. This data set is a fixed size. When the number of archive logs reaches the value of MAXARCH in CSQ6LOGP, or when the BSDS fills up, the oldest archive log information is overwritten.

There are utilities to remove archive log entries from the BSDS, but in general, the BSDS wraps and overlays the oldest archive log record.

When is an archive log needed

You need to back up your page sets regularly. The frequency of backups determines which archive logs are needed in the event of losing a page set.

You need to back up your CF structures regularly. The frequency of backups determines which archive logs are needed in the event of losing data in the CF structure.

The archive log might be needed for recovery. The following information explains when the archive log might be needed, where there are problems with different IBM MQ resources.

Loss of Page set 0
: You must recover your system from your backup and restart the queue manager.

  You need the logs from when the backup was taken, plus up to three active logs.

Loss of any other page set
: You must recover your system from your backup and restart the queue manager.

  You need the logs from when the backup was taken, plus up to three active logs.

All LPARS lose connectivity to a structure, or the structure is unavailable
: Use the RECOVER CFSTRUCT command to read from the last CF backup on the logs.

  If you have been doing frequent backups of the CF, the data should be in active logs.

  You should not need archive logs.

Administration structure rebuild
: If you need to rebuild the administration structure, the information is read from the last checkpoint of the log for each queue manager.

  If a queue manager is not active, another queue manager reads the log.

  You should not need archive logs.

Loss of an SMDS data set
: If you lose an SMDS data set, or the data set gets corrupted, the data set becomes unusable and the status for it is set to FAILED. The CF structure is unchanged.

  In order to restore the SMDS data set, you need to:

  1. Redefine the SMDS data set, and
  2. Fail and then recover the CF structure.

     Issuing the RECOVER CFSTRUCT command twice achieves this process.

     Issuing the command the first time sets the structure state to failed; issuing the command a second time does the actual recovery.

     Note: All non persistent messages on the CF structure will be lost; all persistent messages will be restored.

  You will need the logs from the time the BACKUP CFSTRUCT command was issued, so this might require archive logs.

  If all LPARs lose connectivity to the structure, the structure is re-created, possibly in an alternative CF. Note that your structure CFRM PREFLIST attribute must contain multiple CFs.

  Note: All non persistent messages will be lost; all persistent messages will be re-created by:

  1. Reading the log for the last CF backup
  2. Reading the logs from all queue managers that have used the structure, and
  3. Merging updates since the backup

  You require the logs from all queue managers that have accessed the structure since the last backup (back to the time when the backup was taken) plus the structure backup itself in the log of the queue manager that took the backup.

BSDS

Do you need single or dual BSDS?

If you are using dual active logs you should use dual BSDS.

How big does the BSDS need to be?

The BSDS does not need to be very large, and a primary and secondary of one cylinder should be sufficient.

Planning to increase the maximum addressable log range

z/OS

You can increase the maximum addressable log range by configuring your queue manager to use a larger log relative byte address (RBA).

For an overview of the change to the log RBA for IBM MQ Version 8.0 , see Larger log Relative Byte Address.

If the queue manager is not in a queue-sharing group, you can upgrade the queue manager to CD IBM MQ Version 9.0, enable Version 8.0 new functions, and convert it to use 8 byte log RBA values at any time. Once a queue manager has been converted to use 8 byte log RBA values, it is not possible to revert it to COMPAT mode.

For queue managers in a queue-sharing group, you can upgrade each queue manager in turn to CD IBM MQ Version 9.0 and enable Version 8.0 new function. Once all the queue managers in the group are at IBM MQ Version 8.0 new function mode, you can change each queue manager in turn to use 8 byte log RBA values. It is not essential to change all the queue managers at the same time.

When a queue manager in a queue-sharing group has been converted to use 8 byte log RBA values, other queue managers in the queue-sharing group can use the logs of the converted queue manager, even though they have not yet been converted to use 8 byte log RBA values. This is useful, for example, for peer recovery.

Note: A queue manager that has been converted to use 8 byte log RBA values can read logs that have data written with 6 byte or 8 byte log RBA values. Therefore, its active logs and archive logs can be used to recover page sets and coupling facility (CF) structures.

Undoing the change

The change cannot be backed out.

How long does it take?

The change requires a queue manager restart. Stop the queue manager, run the CSQJUCNV utility against the bootstrap data set (BSDS), or data sets, to create new data sets, rename these bootstrap data sets, and restart the queue manager.

What impact does this have?

- With 8 byte log RBA in use, every write of data to the log data sets has additional bytes. Therefore, for a workload consisting of persistent messages there is a small increase in the amount of data written to the logs.
- Data written to a page set, or coupling facility (CF) structure, is not affected.

Related information:
Implementing the larger log Relative Byte Address

Planning for Managed File Transfer

z/OS

Use this topic as guidance on how you need to set up your system to run Managed File Transfer (MFT). The queue manager needs to be a the same or higher level than the MFT code.

Common configurations

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.

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.

Notes:
1. You can use the same queue manager for transferring data, commands and coordination.
2. This setup, although the simplest, might not be the most efficient because all the workload is on one queue manager.

If you have an existing Managed File Transfer configuration, your command and coordination queue manager might already exist.

If you do not have an existing Managed File Transfer configuration, you can use one queue manager for transferring data, commands, and coordination. Note that even if you do this, it is possible to set up multiple configurations on the same machine.

If you are using multiple queue managers you need to set up channels between the queue managers. You can either do this by using clustering or by using point-to-point connections. Managed File Transfer status and activity can be logged, and can be stored in either a Db2 or Oracle database.

Managed File Transfer is written in Java, with some shell scripts and JCL to configure and operate the program.

Important: You must be familiar with UNIX System Services (USS) in order to configure Managed File Transfer. For example:
- The file directory structure, with names such as `/u/userID/myfile.txt`
- USS commands, for example:
  - `cd` (change directory)
  - `ls` (list)
  - `chmod` (change the file permissions)
  - `chown` (change file ownership or groups which can access the file or directory)

You require the following products in USS to be able to configure and run MFT:

1. Java, for example, in directory `/java/java71_bit64_GA/J7.1_64/`
2. IBM MQ Version 8.0 , for example, in directory `/mqm/V8R0M0`
3. If you want to use Db2 for status and history, you need to install Db2 JDBC libraries, for example, in directory `/db2/db2v10/jdbc/libs`.

Product registration

At startup Managed File Transfer checks the registration in `sys1.parmlib`(IFAPRDxx) concatenation. The following code is an example of how you register MFT:

```
PRODUCT OWNER('IBM CORP')
NAME('WS MQ FILE TRANS')
ID(5655-MFT)
VERSION(*) RELEASE(*) MOD(*)
FEATURENAME('WS MQ FILE TRANS')
STATE(ENABLED)
```

Disk space

You will need 100 MB of DASD for PDSEs and a minimum of 50 MB in USS. If you used trace to diagnose problems, you need additional disk space in USS, for example 50 MB.

Security

You need to identify which user IDs are going to be used for MFT configuration and for MFT operation.

You need to identify the files or queues you transfer, and which user IDs are going to be submitting transfer requests to MFT.

When you customize the agents and logger, you specify the group of users that is allowed to run MFT services, or do MFT administration.

You should set up this group before you start customizing MFT. As MFT uses IBM MQ queues, if you have security enabled in the queue manager, MFT requires access to the following resources:

Table 25. MQADMIN resource class

Name	Access required
QUEUE.SYSTEM.FTE.EVENT.agent_name	Update
QUEUE.SYSTEM.FTE.COMMAND.agent_name	Update
CONTEXT.SYSTEM.FTE.COMMAND.agent_name	Update
QUEUE.SYSTEM.FTE.STATE.agent_name	Update
QUEUE.SYSTEM.FTE.DATA.agent_name	Update
QUEUE.SYSTEM.FTE.REPLY.agent_name	Update
QUEUE.SYSTEM.FTE.AUTHAGT1.agent_name	Update
QUEUE.SYSTEM.FTE.AUTHTRN1.agent_name	Update
QUEUE.SYSTEM.FTE.AUTHOPS1.agent_name	Update
QUEUE.SYSTEM.FTE.AUTHSCH1.agent_name	Update
QUEUE.SYSTEM.FTE.AUTHMON1.agent_name	Update
QUEUE.SYSTEM.FTE.AUTHADM1.agent_name	Update

Table 26. MQQUEUE resource class

Name	Access required
SYSTEM.FTE.AUTHAGT1.agent_name	Update
SYSTEM.FTE.AUTHTRN1.agent_name	Update
SYSTEM.FTE.AUTHOPS1.agent_name	Update
SYSTEM.FTE.AUTHSCH1.agent_name	Update
SYSTEM.FTE.AUTHMON1.agent_name	Update

You can use user sandboxing to determine which parts of the file system the user who requests the transfer can access.

To enable user sandboxing, add the `userSandboxes=true` statement to the *agent.properties* file for the agent that you want to restrict, and add appropriate values to the `MQ_DATA_PATH/mqft/config/coordination_qmgr_name/agents/agent_name/UserSandboxes.xml` file.

See Working with user sandboxes for further information.

This user ID is configured in `UserSandboxes.xml` files.

This XML file has information like user ID, or user ID* and a list of resource that can be used (included), or cannot be used (excluded). You need to define specific user IDs that can access which resources: for example:

Table 27. Example user ID together with access to specific resources

User ID	Access	Include or Exclude	Resource
Admin*	Read	Include	`/home/user/**`
Admin*	Read	Exclude	`/home/user/private/**`
Sysprog	Read	Include	`/home/user/**`
Admin*	Read	Include	Application.reply.queue

Notes:

1. If `type=queue` is specified, the resource is either a queue name, or `queue@qmgr`.
2. If the resource begins with `//`, the resource is a data set; otherwise the resource is a file in USS.
3. The user ID is the user ID from the MQMD structure, so this might not reflect the user ID that actually puts the message.
4. For requests on the local queue manager you can use MQADMIN CONTEXT.* to limit which users can set this value.
5. For requests coming in over a remote queue manager, you have to assume that the distributed queue managers have security enabled to prevent unauthorized setting of the user ID in the MQMD structure.
6. A user ID of SYSPROG1 on a Linux machine, is the same user ID SYSPROG1 for the security checking on z/OS.

How many agents do I need?

The agents do the work in transferring data, and when you make a request to transfer data you specify the name of an agent.

By default an agent can process 25 send and 25 receive requests concurrently. You can configure these processes. See Managed File Transfer configuration options on z/OS for more information.

If the agent is busy then work is queued. The time taken to process a request depends on multiple factors, for example, the amount of data to be sent, the network bandwidth, and the delay on the network.

You might want to have multiple agents to process work in parallel.

You can also control which resources an agent can access, so you might want some agents to work with a limited subset of data.

If you want to process requests with different priority you can use multiple agents and use workload manager to set the priority of the jobs.

Running the agents

Typically the agents are long running processes. The processes can be submitted as jobs that run in batch, or as started tasks.

Planning for channel initiator SMF data

z/OS

You need to plan the implementation of SMF data for the channel initiator (CHINIT).

The CHINIT produces two types of record:
- Statistics data with information about the CHINIT and the tasks within it.
- Channel accounting data with information similar to the DIS CHSTATUS command.

You start collecting statistics data using:
```
/CPF START TRACE(S) class(4)
```

and stop it using
```
/CPF STOP TRACE(S) class(4)
```

You start collecting accounting data using:
```
/CPF START TRACE(A) class(4)
```

and stop it using
```
/CPF STOP TRACE(A) class(4)
```

The SMF records are produced when:
- The time interval in the STATIME ZPARM parameter has elapsed, or if STATIME is zero, on the SMF broadcast. The request to collect SMF data for the CHINIT and the queue manager are synchronized.
- A `STOP TRACE(A) CLASS(4)` or `STOP TRACE(S) CLASS(4)` command is issued, or
- When the CHINIT is shut down. At this point any SMF is written out.

The statistics SMF data normally fits into one SMF record, however, multiple SMF records might be created if a large number of tasks are in use.

Accounting data is gathered for each channel for which it is enabled (using the STATCHL attribute) and normally fits into one SMF record. However, multiple SMF records might be created if a large number of channels are active.

If a channel stops in the interval, accounting data is written to SMF the next time the SMF processing runs. If a client connects, does some work and disconnects, then reconnects and disconnects, there are two sets of channel accounting data produced.

You can control which channels have information written to SMF:
1. By using the STATCHL option on the channel and queue manager.
2. For client channels, note that, you must set STATCHL at the queue manager level.
3. For automatically defined cluster sender channels, you must set STATACLS.

The cost of using the CHINIT SMF data is small. Typically the increase in CPU usage is under a few percent, and often within measurement error.

Before you use this function you need to work with your z/OS systems programmer to ensure that SMF has the capacity for the additional records, and that they change their processes for extracting SMF records to include the new SMF data.

For CHINIT statistics the SMF record type is 115 and sub-type 231.

For CHINIT accounting the SMF record type is 116 and sub-type 10.

You can write your own programs to process this data, or use the SupportPac MP1B that contains a program, MQSMF, for printing the data, and creating data in Comma Separated Values (CSV) suitable for importing into a spread sheet.

If you are experiencing issues with capturing channel initiator SMF data, see Dealing with issues when capturing SMF data for the channel initiator (CHINIT) for further information.
Related information:
Interpreting IBM MQ performance statistics
Troubleshooting channel accounting data

Planning for backup and recovery

z/OS

Developing backup and recovery procedures at your site is vital to avoid costly and time-consuming losses of data. IBM MQ provides means for recovering both queues and messages to their current state after a system failure.

This topic contains the following sections:
- "Recovery procedures" on page 177
- "Tips for backup and recovery" on page 177
- "Recovering page sets" on page 179
- "Recovering CF structures" on page 180
- "Achieving specific recovery targets" on page 181
- "Backup considerations for other products" on page 183
- "Recovery and CICS" on page 183
- "Recovery and IMS" on page 183
- "Preparing for recovery on an alternative site" on page 184
- "Example of queue manager backup activity" on page 184

Recovery procedures

Develop the following procedures for IBM MQ:

- Creating a point of recovery.
- Backing up page sets.
- Backing up CF structures.
- Recovering page sets.
- Recovering from out-of-space conditions (IBM MQ logs and page sets).
- Recovering CF structures.

See Administering IBM MQ for z/OS for information about these.

Become familiar with the procedures used at your site for the following:

- Recovering from a hardware or power failure.
- Recovering from a z/OS component failure.
- Recovering from a site interruption, using off-site recovery.

Tips for backup and recovery

z/OS

Use this topic to understand some backup and recovery tasks.

The queue manager restart process recovers your data to a consistent state by applying log information to the page sets. If your page sets are damaged or unavailable, you can resolve the problem using your backup copies of your page sets (if all the logs are available). If your log data sets are damaged or unavailable, it might not be possible to recover completely.

Consider the following points:

- Periodically take backup copies
- Do not discard archive logs you might need
- Do not change the DDname to page set association

Periodically take backup copies

A *point of recovery* is the term used to describe a set of backup copies of IBM MQ page sets and the corresponding log data sets required to recover these page sets. These backup copies provide a potential restart point in the event of page set loss (for example, page set I/O error). If you restart the queue manager using these backup copies, the data in IBM MQ is consistent up to the point that these copies were taken. Provided that all logs are available from this point, IBM MQ can be recovered to the point of failure.

The more recent your backup copies, the quicker IBM MQ can recover the data in the page sets. The recovery of the page sets is dependent on all the necessary log data sets being available.

In planning for recovery, you need to determine how often to take backup copies and how many complete backup cycles to keep. These values tell you how long you must keep your log data sets and backup copies of page sets for IBM MQ recovery.

When deciding how often to take backup copies, consider the time needed to recover a page set. The time needed is determined by the following:

- The amount of log to traverse.
- The time it takes an operator to mount and remove archive tape volumes.
- The time it takes to read the part of the log needed for recovery.

- The time needed to reprocess changed pages.
- The storage medium used for the backup copies.
- The method used to make and restore backup copies.

In general, the more frequently you make backup copies, the less time recovery takes, but the more time is spent making copies.

For each queue manager, you should take backup copies of the following:
- The archive log data sets
- The BSDS copies created at the time of the archive
- The page sets
- Your object definitions
- Your CF structures

To reduce the risk of your backup copies being lost or damaged, consider:
- Storing the backup copies on different storage volumes to the original copies.
- Storing the backup copies at a different site to the original copies.
- Making at least two copies of each backup of your page sets and, if you are using single logging or a single BSDS, two copies of your archive logs and BSDS. If you are using dual logging or BSDS, make a single copy of both archive logs or BSDS.

Before moving IBM MQ to a production environment, fully test and document your backup procedures.

Backing up your object definitions

Create backup copies of your object definitions. To do this, use the MAKEDEF feature of the COMMAND function of the utility program (described in Using the COMMAND function of CSQUTIL).

You should do this whenever you take backup copies of your queue manager data sets, and keep the most current version.

Backing up your coupling facility structures

If you have set up any queue-sharing groups, even if you are not using them, you must take periodic backups of your CF structures. To do this, use the IBM MQ BACKUP CFSTRUCT command. You can use this command only on CF structures that are defined with the RECOVER(YES) attribute. If any CF entries for persistent shared messages refer to offloaded message data stored in a shared message data set (SMDS) or Db2, the offloaded data is retrieved and backed up together with the CF entries. Shared message data sets should not be backed up separately.

It is recommended that you take a backup of all your CF structures about every hour, to minimize the time it takes to restore a CF structure.

You could perform all your CF structure backups on a single queue manager, which has the advantage of limiting the increase in log use to a single queue manager. Alternatively, you could perform backups on all the queue managers in the queue-sharing group, which has the advantage of spreading the workload across the queue-sharing group. Whichever strategy you use, IBM MQ can locate the backup and perform a RECOVER CFSTRUCT from any queue manager in the queue-sharing group. The logs of all the queue managers in the queue-sharing group need to be accessed to recover the CF structure.

Backing up your message security policies

If you are using Advanced Message Security to create a backup of your message security policies, create a backup using the message security policy utility (CSQ0UTIL) to run **`dspmqspl`** with the -export parameter, then save the policy definitions that are output to the EXPORT DD.

You should create a backup of your message security policies whenever you take backup copies of your queue manager data sets, and keep the most current version.

Do not discard archive logs you might need

IBM MQ might need to use archive logs during restart. You must keep sufficient archive logs so that the system can be fully restored. IBM MQ might use an archive log to recover a page set from a restored backup copy. If you have discarded that archive log, IBM MQ cannot restore the page set to its current state. When and how you discard archive logs is described in Discarding archive log data sets.

You can use the `/cpf DIS USAGE TYPE(ALL)` command to display the log RBA, and log range sequence number (LRSN) that you need to recover your queue manager's page sets and the queue-sharing group's structures. You should then use the print log map utility (CSQJU004) to print bootstrap data set (BSDS) information for the queue manager to locate the logs containing the log RBA.

For structures, you need to run the CSQJU004 utility on each queue manager in the queue-sharing group to locate the logs containing the LRSN. You need these logs and any later logs to be able to recover the page sets and structures.

Do not change the DDname to page set association

IBM MQ associates page set number 00 with DDname CSQP0000, page set number 01 with DDname CSQP0001, and so on, up to CSQP0099. IBM MQ writes recovery log records for a page set based on the DDname that the page set is associated with. For this reason, you must not move page sets that have already been associated with a PSID DDname.

Recovering page sets

z/OS

Use this topic to understand the factors involved when recovering pages sets, and how to minimize restart times.

A key factor in recovery strategy concerns the time for which you can tolerate a queue manager outage. The total outage time might include the time taken to recover a page set from a backup, or to restart the queue manager after an abnormal termination. Factors affecting restart time include how frequently you back up your page sets, and how much data is written to the log between checkpoints.

To minimize the restart time after an abnormal termination, keep units of work short so that, at most, two active logs are used when the system restarts. For example, if you are designing an IBM MQ application, avoid placing an `MQGET` call that has a long wait interval between the first in-syncpoint MQI call and the commit point because this might result in a unit of work that has a long duration. Another common cause of long units of work is batch intervals of more than 5 minutes for the channel initiator.

You can use the DISPLAY THREAD command to display the RBA of units of work and to help resolve the old ones.

How often must you back up a page set?

Frequent page set backup is essential if a reasonably short recovery time is required. This applies even when a page set is very small or there is a small amount of activity on queues in that page set.

If you use persistent messages in a page set, the backup frequency should be in hours rather than days. This is also the case for page set zero.

To calculate an approximate backup frequency, start by determining the target total recovery time. This consists of the following:

1. The time taken to react to the problem.
2. The time taken to restore the page set backup copy.

   If you use SnapShot backup/restore, the time taken to perform this task is a few seconds. For information about SnapShot, see the *DFSMSdss Storage Administration Guide*.
3. The time the queue manager requires to restart, including the additional time needed to recover the page set.

   This depends most significantly on the amount of log data that must be read from active and archive logs since that page set was last backed up. All such log data must be read, in addition to that directly associated with the damaged page set.

   Note: When using *fuzzy backup* (where a snapshot is taken of the logs and page sets while a unit of work is active), it might be necessary to read up to three additional checkpoints, and this might result in the need to read one or more additional logs.

When deciding on how long to allow for the recovery of the page set, the factors that you need to consider are:

- The rate at which data is written to the active logs during normal processing depends on how messages arrive in your system, in addition to the message rate.

  Messages received or sent over a channel result in more data logging than messages generated and retrieved locally.
- The rate at which data can be read from the archive and active logs.

  When reading the logs, the achievable data rate depends on the devices used and the total load on your particular DASD subsystem.

  With most tape units, it is possible to achieve higher data rates for archived logs with a large block size. However, if an archive log is required for recovery, all the data on the active logs must be read also.

Recovering CF structures

z/OS

Use this topic to understand the recovery process for CF structures.

At least one queue manager in the queue-sharing group must be active to process a RECOVER CFSTRUCT command. CF structure recovery does not affect queue manager restart time, because recovery is performed by an already active queue manager.

The recovery process consists of two logical steps that are managed by the RECOVER CFSTRUCT command:

1. Locating and restoring the backup.
2. Merging all the logged updates to persistent messages that are held on the CF structure from the logs of all the queue managers in the queue-sharing group that have used the CF structure, and applying the changes to the backup.

The second step is likely to take much longer because a lot of log data might need to be read. You can reduce the time taken if you take frequent backups, or if you recover multiple CF structures at the same time, or both.

The queue manager performing the recovery locates the relevant backups on all the other queue managers' logs using the data in Db2 and the bootstrap data sets. The queue manager replays these backups in the correct time sequence across the queue-sharing group, from just before the last backup through to the point of failure.

The time it takes to recover a CF structure depends on the amount of recovery log data that must be replayed, which in turn depends on the frequency of the backups. In the worst case, it takes as long to read a queue manager's log as it did to write it. So if, for example, you have a queue-sharing group containing six queue managers, an hour's worth of log activity could take six hours to replay. In general it takes less time than this, because reading can be done in bulk, and because the different queue manager's logs can be read in parallel. As a starting point, we recommend that you back up your CF structures every hour.

All queue managers can continue working with non-shared queues and queues in other CF structures while there is a failed CF structure. If the administration structure has also failed, at least one of the queue managers in the queue-sharing group must be started before you can issue the RECOVER CFSTRUCT command.

Backing up CF structures can require considerable log writing capacity, and can therefore impose a large load on the queue manager doing the backup. Choose a lightly loaded queue manager for doing backups; for busy systems, add an additional queue manager to the queue-sharing group and dedicate it exclusively for doing backups.

Achieving specific recovery targets

z/OS

Use this topic for guidance on how you can achieve specific recovery target times by adjusting backup frequency.

If you have specific recovery targets to achieve, for example, completion of the queue manager recovery and restart processing in addition to the normal startup time within *xx* seconds, you can use the following calculation to estimate your backup frequency (in hours):

```
Formula (A)
                    Required restart time * System recovery log read rate
                          (in secs)                  (in MB/sec)
Backup frequency  = -----------------------------------------------------
   (in hours)                Application log write rate (in MB/hour)
```

Note: The examples given next are intended to highlight the need to back up your page sets frequently. The calculations assume that most log activity is derived from a large number of persistent messages. However, there are situations where the amount of log activity is not easily calculated. For example, in a queue-sharing group environment, a unit of work in which shared queues are updated in addition to other resources might result in UOW records being written to the IBM MQ log. For this reason, the Application log write rate in Formula (A) can be derived accurately only from the observed rate at which the IBM MQ logs fill.

For example, consider a system in which IBM MQ MQI clients generate a total load of 100 persistent messages a second. In this case, all messages are generated locally.

If each message is of user length 1 KB, the amount of data logged each hour is approximately:

```
100 * (1 + 1.3) KB * 3600 = approximately 800 MB

where
      100           = the message rate a second
      (1 + 1.3) KB  = the amount of data logged for
                      each 1 KB of persistent messages
```

Consider an overall target recovery time of 75 minutes. If you have allowed 15 minutes to react to the problem and restore the page set backup copy, queue manager recovery and restart must then complete within 60 minutes (3600 seconds) applying formula (A). Assuming that all required log data is on RVA2-T82 DASD, which has a recovery rate of approximately 2.7 MB a second, this necessitates a page set backup frequency of at least every:

```
3600 seconds * 2.7 MB a second / 800 MB an hour = 12.15 hours
```

If your IBM MQ application day lasts approximately 12 hours, one backup each day is appropriate. However, if the application day lasts 24 hours, two backups each day is more appropriate.

Another example might be a production system in which all the messages are for request-reply applications (that is, a persistent message is received on a receiver channel and a persistent reply message is generated and sent down a sender channel).

In this example, the achieved batch size is one, and so there is one batch for every message. If there are 50 request replies a second, the total load is 100 persistent messages a second. If each message is 1 KB in length, the amount of data logged each hour is approximately:

```
50((2 * (1+1.3) KB) + 1.4 KB + 2.5 KB) * 3600 = approximately 1500 MB

where:
 50                 = the message pair rate a second
 (2 * (1 + 1.3) KB) = the amount of data logged for each message pair
 1.4 KB             = the overhead for each batch of messages
                      received by each channel
 2.5 KB             = the overhead for each batch of messages sent
                      by each channel
```

To achieve the queue manager recovery and restart within 30 minutes (1800 seconds), again assuming that all required log data is on RVA2-T82 DASD, this requires that page set backup is carried out at least every:

```
1800 seconds * 2.7 MB a second / 1500 MB an hour = 3.24 hours
```

Periodic review of backup frequency

Monitor your IBM MQ log usage in terms of MB an hour. Periodically perform this check and amend your page set backup frequency if necessary.

Backup considerations for other products

z/OS

If you are using IBM MQ with CICS or IMS then you must also consider the implications for your backup strategy with those products. The data facility hierarchical storage manager (DFHSM) manages data storage, and can interact with the storage used by IBM MQ.

Backup and recovery with DFHSM

The data facility hierarchical storage manager (DFHSM) does automatic space-availability and data-availability management among storage devices in your system. If you use it, you need to know that it moves data to and from the IBM MQ storage automatically.

DFHSM manages your DASD space efficiently by moving data sets that have not been used recently to alternative storage. It also makes your data available for recovery by automatically copying new or changed data sets to tape or DASD backup volumes. It can delete data sets, or move them to another device. Its operations occur daily, at a specified time, and allow for keeping a data set for a predetermined period before deleting or moving it.

You can also perform all DFHSM operations manually. The *Data Facility Hierarchical Storage Manager User's Guide* explains how to use the DFHSM commands. If you use DFHSM with IBM MQ, note that DFHSM does the following:

- Uses cataloged data sets.
- Operates on page sets and logs.
- Supports VSAM data sets.

Recovery and CICS

The recovery of CICS resources is not affected by the presence of IBM MQ. CICS recognizes IBM MQ as a non-CICS resource (or external resource manager), and includes IBM MQ as a participant in any syncpoint coordination requests using the CICS resource manager interface (RMI). For more information about CICS recovery, see the *CICS Recovery and Restart Guide*. For information about the CICS resource manager interface, see the *CICS Customization Guide*.

Recovery and IMS

IMS recognizes IBM MQ as an external subsystem and as a participant in syncpoint coordination. IMS recovery for external subsystem resources is described in the *IMS Customization Guide*.

Preparing for recovery on an alternative site

z/OS

If a total loss of an IBM MQ computing center, you can recover on another IBM MQ system at a recovery site.

To recover an IBM MQ system at a recovery site, you must regularly back up the page sets and the logs. As with all data recovery operations, the objectives of disaster recovery are to lose as little data, workload processing (updates), and time as possible.

At the recovery site:

- The recovery IBM MQ queue manager **must** have the same name as the lost queue manager.
- Ensure the system parameter module used on the recovery queue manager contains the same parameters as the lost queue manager.

The process for disaster recovery is described in the Administering IBM MQ for z/OS.

Example of queue manager backup activity

z/OS

This topic shows as an example of queue manager backup activity.

When you plan your queue manager backup strategy, a key consideration is retention of the correct amount of log data. Managing the logs describes how to determine which log data sets are required, by reference to the system recovery RBA of the queue manager. IBM MQ determines the system recovery RBA using information about the following:

- Currently active units of work.
- Page set updates that have not yet been flushed from the buffer pools to disk.
- CF structure backups, and whether this queue manager's log contains information required in any recovery operation using them.

You must retain sufficient log data to be able to perform media recovery. While the system recovery RBA increases over time, the amount of log data that must be retained only decreases when subsequent backups are taken. CF structure backups are managed by IBM MQ, and so are taken into account when reporting the system recovery RBA. This means that in practice, the amount of log data that must be retained only reduces when page set backups are taken.

Figure 57 on page 185 shows an example of the backup activity on a queue manager that is a member of a queue-sharing group, how the recovery RBA varies with each backup, and how that affects the amount of log data that must be retained. In the example the queue manager uses local and shared resources: page sets, and two CF structures, STRUCTURE1 and STRUCTURE2.

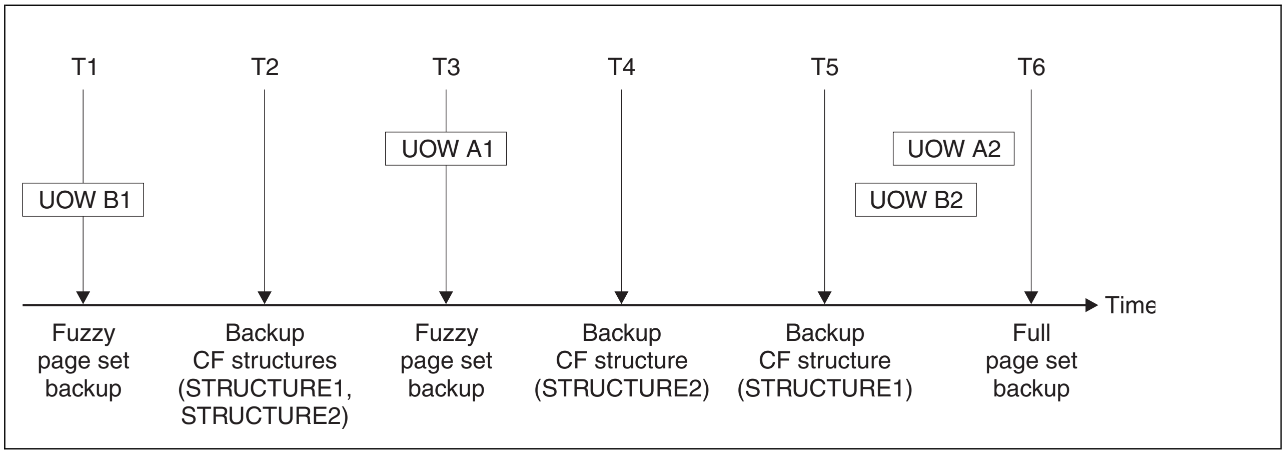

T1
T2
T3
T4
T5
T6
UOW B1
UOW A1
UOW A2
UOW B2
Time
Fuzzy page set backup
Backup CF structures (STRUCTURE1, STRUCTURE2)
Fuzzy page set backup
Backup CF structure (STRUCTURE2)
Backup CF structure (STRUCTURE1)
Full page set backup

Figure 57. Example of queue manager backup activity.

Example of queue manager backup activity.

This is what happens at each point in time:

Point in time T1

A fuzzy backup is created of your page sets, as described in How to back up and recover page sets.

The system recovery RBA of the queue manager is the lowest of the following:

- The recovery RBAs of the page sets being backed up at this point.
- The lowest recovery RBA required to recover the CF application structures. This relates to the recovery of backups of STRUCTURE1 and STRUCTURE2 created earlier.
- The recovery RBA for the oldest currently active unit of work within the queue manager (UOWB1).

The system recovery RBA for this point in time is given by messages issued by the DISPLAY USAGE command, which is part of the fuzzy backup process.

Point in time T2

Backups of the CF structures are created. CF structure STRUCTURE1 is backed up first, followed by STRUCTURE2.

The amount of log data that must be retained is unchanged, because the same data as determined from the system recovery RBA at T1 is still required to recover using the page set backups taken at T1.

Point in time T3

Another fuzzy backup is created.

The system recovery RBA of the queue manager is the lowest of the following:

- The recovery RBAs of the page sets being backed up at this point.
- The lowest recovery RBA required to recover CF structure STRUCTURE1, because STRUCTURE1 was backed up before STRUCTURE2.
- The recovery RBA for the oldest currently active unit of work within the queue manager (UOWA1).

The system recovery RBA for this point in time is given by messages issued by the DISPLAY USAGE command, which is part of the fuzzy backup process.

You can now reduce the log data retained, as determined by this new system recovery RBA.

Point in time T4

A backup is taken of CF structure STRUCTURE2. The recovery RBA for the recovery of the oldest required CF structure backup relates to the backup of CF structure STRUCTURE1, which was backed up at time T2.

The creation of this CF structure backup has no effect on the amount of log data that must be retained.

Point in time T5

A backup is taken of CF structure STRUCTURE1. The recovery RBA for recovery of the oldest required CF structure backup now relates to recovery of CF structure STRUCTURE2, which was backed up at time T4.

The creation of this CF structure backup has no effect on amount of log data that must be retained.

Point in time T6

A full backup is taken of your page sets as described in How to back up and recover page sets.

The system recovery RBA of the queue manager is the lowest of the following:

- The recovery RBAs of the page sets being backed up at this point.
- The lowest recovery RBA required to recover the CF structures. This relates to recovery of CF structure STRUCTURE2.
- The recovery RBA for the oldest currently active unit of work within the queue manager. In this case, there are no current units of work.

The system recovery RBA for this point in time is given by messages issued by the DISPLAY USAGE command, which is part of the full backup process.

Again, the log data retained can be reduced, because the system recovery RBA associated with the full backup is more recent.

Planning your z/OS UNIX or UNIX System Services environment

z/OS

Certain processes within the IBM MQ queue manager (MSTR) and the channel initiator (CHIN) use z/OS UNIX or UNIX System Services (USS) for their normal processing. Plan your configuration if you do not want to use the default USS configuration.

No special action or customization is necessary in order for IBM MQ to use UNIX services as long as a system-wide default OMVS segment has been set up.

Users who do not want IBM MQ to invoke USS, using the guest or default UID and OMVS segment, need only model a new OMVS segment based on the default segment, as IBM MQ requires no special permissions, and does not run within UNIX as a superuser.

Note: Although the MSTR and CHIN jobs make use of USS facilities (for example, to interface with TCP/IP services), they do not need to access any of the content of the USS filesystem shipped by IBM MQ. As a result, the MSTR and CHIN jobs do not require any configuration to specify the path for the USS filesystem.

The content of the IBM MQ directory in the USS filesystem is used by applications connecting to IBM MQ. For example, applications using the IBM MQ classes for Java or IBM MQ classes for JMS interfaces.

See the following topics for the relevant configuration instructions:

- Environment variables relevant to IBM MQ classes for Java
- IBM MQ classes for Java libraries

- Setting environment variables
- Configuring the Java Native Interface (JNI) libraries

Planning your z/OS TCP/IP environment

z/OS

To get the best throughput through your network, you must use TCP/IP send and receive buffers with a size of 64 KB, or greater. With this size, the system optimizes its buffer sizes.

See Dynamic right sizing for high latency networks for more information.

You can check your system buffer size by using the following Netstat command, for example:

```
TSO NETSTAT ALL (CLIENT csq1CHIN
```

The results display much information, including the following two values:

```
ReceiveBufferSize: 0000065536
SendBufferSize: 0000065536
```

65536 is 64 KB. If your buffer sizes are less than 65536, you must work with your network team to increase the **TCPSENDBFRSIZE** and **TCPRCVBUFRSIZE** values in the `PROFILE DDName` in the TCPIP procedure. For example, you might use the following command:

```
TCPCONFIG TCPSENDBFRSZE 65536 TCPRCVBUFRSIZE 65536
```

If you are unable to change your system-wide **TCPSENDBFRSIZE** or **TCPRCVBUFRSIZE** settings, contact your IBM Software Support center.

Planning to use the IBM MQ Console and REST API on z/OS

z/OS V 9.0.2

The IBM MQ Console and REST API are WebSphere Application Server Liberty (Liberty) applications running in a server known as `mqweb`. The `mqweb` server runs as a started task. The MQ Console allows a web browser to be used to administer queue managers, and the REST API provides a simple programmatic interface for applications to do queue manager administration. V 9.0.4 From Version 9.0.4, the REST API also performs messaging.

The MQ Console and REST API can only directly interact with queue managers running at the same Version, Release, and Modification (VRM) . For example, the MQ Console and REST API shipped with Version 9.0.3 can only interact with local queue managers at Version 9.0.3, and the MQ Console and REST API shipped with Version 9.0.4 can only interact with local queue managers at Version 9.0.4.

Migration

If you have only one queue manager, you can run the `mqweb` server as a started task called MQWEB, and change the libraries it uses when you migrate your queue manager.

If you have more than one queue manager, during migration you can start a Version 9.0.3 and a Version 9.0.4 `mqweb` server, using started tasks named MQWB0903 and MQWB0904 respectively.

Note that these are example names for the started tasks, you can use any name you want.

When you migrate the queue managers, the queue managers become available in the Version 9.0.4 `mqweb` server.

Once you have migrated all the queue managers to Version 9.0.4, you can delete the Version 9.0.3 mqweb server.

V 9.0.4 In Version 9.0.4 and later, you can administer queue managers at a different version from the mqweb server by configuring a gateway queue manager. For more information, see Remote administration using the REST API.

Note: You still need at least one queue manager at the same version as the mqweb server, to act as the gateway queue manager.

HTTP ports

The mqweb server uses at least one, and up to two ports for HTTP:
- One for HTTPS; this value defaults to 9443
- One for HTTP; HTTP is not enabled by default, but if enabled, will use port 9080 by default

Notes:
1. If the default ports are already in use, you need to allocate other ports.
2. If you have more than one mqweb server running simultaneously for more than one version of IBM MQ, you need to allocate separate ports for each version.

You can use the command `tso netstat tcp tcpip portlist (port 9080)` to display information about a port - in this case, 9080.

Installation and configuration files

You need to install the IBM MQ for z/OS Unix System Services Web Components feature, which will install the files needed to run the mqweb server in USS. You need to be familiar with USS to be able to configure and manage the mqweb server.

If you have installed IBM MQ on one system, and run IBM MQ on a different system, you should copy the IBM MQ ZFS created during the installation to the processing system, and mount it read only.

If the directory where the IBM MQ files are to be stored is `/mqm/v904`, you can use the `df -P/mqm/v904` command to display the free and available space. Note that IBM MQ requires 646516 one kilobyte pages.

You need to create, and decide upon the location for, a Liberty user directory when you configure the mqweb server. This directory contains configuration and log files, and the location can be something similar to `/var/mqm/web/mqweb904`.

Security

The MQ Console and REST API, issue commands to the queue manager using the PCF interface. The user ID of the mqweb server needs authority to issue IBM MQ commands and access certain queues. These details are described in IBM MQ Console - required command security profiles.

If you are using the SAF interface for authentication, you need a Liberty Angel process running. If you already have a Liberty Profile running on the LPAR, the Angel process might already be running. See Enabling z/OS authorized services on Liberty for z/OS, and Configuring the Liberty Angel process and z/OS authorized services for more information.

The mqweb started task user ID needs:
- A uid to be able to use USS.
- Access to the `hlq.SCSQAUTH` and `hlq.SCSQANL*` data sets in the IBM MQ installation.

- Read access to the IBM MQ installation files in USS.
- Read and write access to the Liberty user directory.
- Access to various security profiles. See Configuring System Authorization Facility interface for more information.

The Liberty server use the default security prefix BBGZDFLT for its security definitions. If this is already being used, you might want to use a different prefix.

Installing and uninstalling IBM MQ

Before you start installing IBM MQ, consider how you want to use it. Use these topics to help you to prepare for installation, install the product, and verify the installation. There is also information to help you to uninstall the product.

About this task

To get started with installing IBM MQ, see the topics for the platform, or platforms, that your enterprise uses. For concepts and considerations relating to installation, see "IBM MQ installation overview" on page 192.

You can also apply and remove maintenance to IBM MQ. See Maintenance tasks in the Maintaining and migrating section.

Attention: The information in this section applies to both Continuous Delivery (CD) and Long Term Support (LTS) releases.

Any information that applies specifically to an LTS or CD release is marked with the appropriate icon.

Procedure

1. To find information on installing IBM MQ, see the appropriate sections for the platform, or platforms, that your enterprise uses:
   - AIX "Installing and uninstalling IBM MQ on AIX" on page 215
   - NSS Client "Installing and uninstalling IBM MQ client for HP Integrity NonStop Server" on page 249
   - HP-UX "Installing and uninstalling IBM MQ on HP-UX" on page 263
   - Linux "Installing IBM MQ on Linux using rpm" on page 340
   - Solaris "Installing and uninstalling IBM MQ on Solaris" on page 398
   - Windows "Installing and uninstalling IBM MQ on Windows" on page 435
   - z/OS "Installing IBM MQ for z/OS" on page 547
2. To find out about concepts and considerations relating to installation, see "IBM MQ installation overview" on page 192.

IBM MQ installation overview

An overview of concepts and considerations for installing IBM MQ, with links to instructions on how to install, verify, and uninstall IBM MQ on each of the supported platforms.

Related concepts:

"Multiple installations on UNIX, Linux, and Windows" on page 200
On UNIX, Linux, and Windows, it is possible to have more than one copy of IBM MQ on a system.

"Installation considerations for MQ Telemetry" on page 538
From IBM WebSphere MQ Version 7.1, MQ Telemetry is a component of the main IBM MQ product, and is no longer a separate plugin. You can choose to install MQ Telemetry when you first install IBM MQ, or when you modify an existing IBM MQ installation. To get a set of client libraries that help you write messaging applications for telemetry, download the IBM Messaging Telemetry Clients SupportPac.

Related tasks:

Installing Advanced Message Security
Use the information for your platform to guide you through installing the Advanced Message Security (AMS) component.

Related information:

Maintaining and migrating

Managed File Transfer product options

IBM MQ components and features

You can select the components or features that you require when you install IBM MQ.

Important: Ensure that your enterprise has the correct license, or licenses, for the components that you are going to install. For more information, see "License requirements" on page 194 and IBM MQ license information.

Also review the information on hardware and software requirements for the platform on which you are planning to install IBM MQ. For more information, see "Where to find product requirements and support information" on page 195.

Multi

Installation of IBM MQ on Multiplatforms

IBM MQ can be installed as a server or a client. The installation images can be downloaded (see "Installation with a download image" on page 211), or IBM MQ can be installed from a DVD.

An IBM MQ server is an installation of one or more queue managers that provide queuing services to one or more clients. All the IBM MQ objects, for example queues, exist only on the queue manager machine (the IBM MQ server machine), and not the client. An IBM MQ server can also support local IBM MQ applications.

An IBM MQ MQI client is a component that allows an application running on one system to communicate with a queue manager running on another system. The output from the call is sent back to the client, which passes it back to the application.

For detailed explanations of all the components that you can install, see:

- AIX "IBM MQ components for AIX" on page 216
- HP-UX "IBM MQ components for HP-UX" on page 264
- IBM i "IBM MQ components for IBM i" on page 298
- Linux "IBM MQ rpm components for Linux systems" on page 341

- Linux “IBM MQ Debian components for Linux Ubuntu systems” on page 364
- Solaris “IBM MQ components for Solaris systems” on page 398
- Windows “IBM MQ features for Windows systems” on page 436

For information about how to install IBM MQ on each supported platform, see the links in the following table:

Table 28. Where to find IBM MQ installation information for each platform

Platform	IBM MQ server	IBM MQ client
AIX AIX	“Installing IBM MQ server on AIX” on page 225	“Installing an IBM MQ client on AIX” on page 231
NSS Client HP Integrity NonStop Server	Not applicable	“Installing and uninstalling IBM MQ client for HP Integrity NonStop Server” on page 249
HP-UX HP-UX	“Installing IBM MQ server on HP-UX” on page 274	“Installing an IBM MQ client on HP-UX” on page 280
IBM i IBM i	“Installing IBM MQ server on IBM i” on page 303	“Installing an IBM MQ client on IBM i” on page 316
Linux Linux	“Installing IBM MQ server on Linux” on page 344	“Installing an IBM MQ client on Linux” on page 355
V 9.0.2 Linux Linux	“Installing an IBM MQ server on Linux Ubuntu using Debian packages” on page 368	“Installing an IBM MQ client on Linux Ubuntu using Debian packages” on page 372
Solaris Solaris	“Installing IBM MQ server on Solaris” on page 410	“Installing an IBM MQ client on Solaris” on page 416
Windows Windows	“Installing IBM MQ server on Windows” on page 453	“Installing an IBM MQ client on Windows” on page 481

Multi

Installing IBM MQ clients and servers on the same system

It is possible to have both a server and a client installation on the same system. If you install from a DVD, to install an IBM MQ client on a system that is already running an IBM MQ server, you must use the appropriate Server DVD. You can use a Client DVD to install an IBM MQ client only on a system that is not running an IBM MQ server.

If you install an IBM MQ client from a Client DVD and later decide to install the IBM MQ server on the same system, you must first remove all the client components from the system, then use the appropriate Server DVD to install both the server and client components. You cannot install an IBM MQ server on a system that already has client components installed from a Client DVD.

Remember that even if your client and server are installed on the same system, you must still define the MQI channel between them. See Defining MQI channels for details.

Advanced Message Security, Managed File Transfer and MQ Telemetry

Advanced Message Security, Managed File Transfer and MQ Telemetry are separately installed components of IBM MQ. Make sure that you purchase a license for using IBM MQ Advanced before installing any of these components (see IBM MQ license information).

z/OS

Installation of IBM MQ on z/OS

For information on installation options for IBM MQ for z/OS, see "Installing IBM MQ for z/OS" on page 547.

V 9.0.0 For information on installation options for IBM MQ Advanced for z/OS, see "Installing IBM MQ Advanced for z/OS" on page 560.

V 9.0.0 For information on installation options for IBM MQ Advanced for z/OS, Value Unit Edition, see "Installing IBM MQ Advanced for z/OS, Value Unit Edition" on page 561.

Related concepts:

"Installation location on Multiplatforms" on page 198
You can install IBM MQ into the default location. Alternatively, you can install into a custom location during the installation process. The location where IBM MQ is installed is known as the *MQ_INSTALLATION_PATH*.

"Primary installation on UNIX, Linux, and Windows" on page 202
On systems that support multiple installations of IBM MQ (UNIX, Linux, and Windows), the primary installation is the one to which IBM MQ system-wide locations refer. Having a primary installation is optional, but convenient.

"Installation name on UNIX, Linux, and Windows" on page 197
Each installation of IBM MQ on UNIX, Linux, and Windows, has a unique identifier known as an installation name. The installation name is used to associate things such as queue managers and configuration files with an installation.

Related tasks:

Multi "Installing IBM MQ Advanced for Multiplatforms" on page 519
Installation tasks associated with IBM MQ Advanced for Multiplatforms are grouped in this section.

License requirements

You must have purchased sufficient licenses for your installation. The details of the license agreement is stored on your system at installation time so that you can read it at any time. IBM MQ supports IBM License Metric Tool (ILMT).

Important: Ensure that your enterprise has the correct license, or licenses, for the components that you are going to install. See IBM MQ license information for more details.

License files

At installation, the license agreement files are copied into the `/licenses` directory under the *MQ_INSTALLATION_PATH*. You can read them at any time.

IBM i On IBM i, you can use the WRKSFWAGR command to view the software licenses.

ILMT

ILMT automatically detects IBM MQ, if you are using it, and checks with it each time a queue manager is started. You do not need to take any further action. You can install ILMT before or after IBM MQ.

The automatic detection applies to both the IBM MQ server and IBM MQ Java products.

Related concepts:

UNIX Linux "Hardware and software requirements on Linux systems" on page 331
Before you install IBM MQ , check that your system meets the hardware and operating system software requirements for the particular components you intend to install.

IBM i "Hardware and software requirements on IBM i systems" on page 299
Check that the server environment meets the prerequisites for installing IBM MQ for IBM i. Check the product readme files and install missing prerequisite software supplied on the server CD.

Windows "Hardware and software requirements on Windows systems" on page 445
Check that the server environment meets the prerequisites for installing IBM MQ for Windows and install any prerequisite software that is missing from your system from the server DVD.

Related tasks:

"Checking requirements on Windows" on page 444
Before you install IBM MQ on Windows, you must check for the latest information and system requirements.

Where to find product requirements and support information

Before you install IBM MQ, you must check for the latest information and system requirements.

You can consult the following sources to check that you have the information that you need to help you with planning your installation, including information on hardware and software requirements:

IBM MQ System Requirements website
For details of hardware and software requirements on all supported platforms, see System Requirements for IBM MQ. From IBM MQ Version 8.0, you can use the Software Product Compatibility Reports (SPCR) tool to find information on supported operating systems, system requirements, prerequisites, and optional supported software. For more information about the SPCR tool and links to reports for each supported platform, see the System Requirements for IBM MQ Version 9.0 web page.

Product readme file
The product readme file includes information about last minute changes and known problems and workarounds. The latest version is available at the IBM MQ, WebSphere MQ, and MQSeries® product readmes web page. Always check to see that you have the latest version of the product readme file.

Support information
The IBM MQ support web page is regularly updated with the latest product support information. For example, if you are migrating from an earlier version, look under the heading *Solve a problem* for the document *Problems and solutions when migrating*.

Related concepts:

"IBM MQ installation overview" on page 192
An overview of concepts and considerations for installing IBM MQ, with links to instructions on how to install, verify, and uninstall IBM MQ on each of the supported platforms.

AIX "Hardware and software requirements on AIX systems" on page 219
Before you install IBM MQ, check that your system meets the hardware and operating system software requirements for the particular components you intend to install.

NSS Client "Hardware and software requirements on HP Integrity NonStop Server systems" on page 250
Check that the server environment meets the prerequisites for installing the IBM MQ client for HP Integrity NonStop Server. Check the product readme files and install missing prerequisite software supplied on the server CD.

HP-UX "Hardware and software requirements on HP-UX systems" on page 266
Before you install IBM MQ , check that your system meets the hardware and operating system software requirements for the particular components you intend to install.

IBM i "Hardware and software requirements on IBM i systems" on page 299
Check that the server environment meets the prerequisites for installing IBM MQ for IBM i. Check the product readme files and install missing prerequisite software supplied on the server CD.

Linux "Hardware and software requirements on Linux systems" on page 331
Before you install IBM MQ , check that your system meets the hardware and operating system software requirements for the particular components you intend to install.

Solaris "Hardware and software requirements on Solaris systems" on page 403
Before you install IBM MQ , check that your system meets the hardware and operating system software requirements for the particular components you intend to install.

Windows "Hardware and software requirements on Windows systems" on page 445
Check that the server environment meets the prerequisites for installing IBM MQ for Windows and install any prerequisite software that is missing from your system from the server DVD.

Related tasks:

z/OS "Installing IBM MQ for z/OS" on page 547
Installation tasks that are associated with installing IBM MQ on z/OS systems are grouped in this section.

Related information:

IBM MQ maintenance tasks

Planning considerations for installation on Multiplatforms

Multi

Before you install IBM MQ, you must choose which components to install and where to install them. You must also make some platform-specific choices.

Before you start installing, consider how you want to use IBM MQ and review the information in this section, and also the information in the general Planning section.

NSS Client Note that HP Integrity NonStop Server uses a different installer from the other platforms, and a multi-installation layout.

When planning your installation, make sure that you check the hardware and software requirements for your system. For more information, see "Where to find product requirements and support information" on page 195.

Note: z/OS This information is about planning an installation on IBM MQ for Multiplatforms. For information about planning an installation on z/OS, see “Planning to install IBM MQ for z/OS” on page 550.

Installation name on UNIX, Linux, and Windows

ULW

Each installation of IBM MQ on UNIX, Linux, and Windows, has a unique identifier known as an installation name. The installation name is used to associate things such as queue managers and configuration files with an installation.

You can choose the installation name and make it meaningful to you. For example, you might call a test system *testMQ*.

If you do not specify an installation name when the product is installed, a default installation name is automatically assigned. For the first installation, this name is *Installation1*. For the second installation, the name is *Installation2*, and so on. The installation name *Installation0* is reserved for an installation of IBM WebSphere MQ Version 7.0.1. The installation name cannot be changed after the product is installed.

UNIX Linux On UNIX and Linux systems, the first IBM MQ installation is automatically given an installation name of *Installation1*. For subsequent installations, you can use the **`crtmqinst`** command to set the installation name before installing the product.

Windows On Windows systems, you can choose the installation name during the installation process.

The installation name can be up to 16 bytes and must be a combination of alphabetic and numeric characters in the ranges a-z, A-Z, and 0-9. You cannot use blank characters. The installation name must be unique, regardless of whether uppercase or lowercase characters are used. For example, the names `INSTALLATIONNAME` and `InstallationName` are not unique.

You can find out what installation name is assigned to an installation in a particular location using the **`dspmqinst`** command.

Installation descriptions

Each installation can also have an installation description. This description can give more detailed information about an installation in cases where the installation name cannot provide enough information. These descriptions can be up to 64 single-byte characters, or 32 double-byte characters. The default installation description is blank. You can set the installation description using the **`setmqinst`** command.

Related concepts:

"Planning considerations for installation on Multiplatforms" on page 196
Before you install IBM MQ, you must choose which components to install and where to install them. You must also make some platform-specific choices.

"Primary installation on UNIX, Linux, and Windows" on page 202
On systems that support multiple installations of IBM MQ (UNIX, Linux, and Windows), the primary installation is the one to which IBM MQ system-wide locations refer. Having a primary installation is optional, but convenient.

"Installation location on Multiplatforms"
You can install IBM MQ into the default location. Alternatively, you can install into a custom location during the installation process. The location where IBM MQ is installed is known as the *MQ_INSTALLATION_PATH*.

"IBM MQ components and features" on page 192
You can select the components or features that you require when you install IBM MQ.

Related information:

dspmqinst

setmqinst

crtmqinst

Installation location on Multiplatforms

Multi

You can install IBM MQ into the default location. Alternatively, you can install into a custom location during the installation process. The location where IBM MQ is installed is known as the *MQ_INSTALLATION_PATH*.

Default location

The default location for the IBM MQ product code is shown in the following table:

Table 29. Installation location of IBM MQ

Platform	Installation location
AIX AIX	`/usr/mqm`
Linux HP-UX Solaris Linux , HP-UX, and Solaris	`/opt/mqm`
IBM i IBM i	`/QIBM/ProdData/mqm`
Windows Windows systems	`C:\Program Files\IBM\MQ`
Windows Windows data directories	`C:\ProgramData\IBM\MQ`

Important: Windows For Windows installations, the directories are as stated, unless there is a previous installation of the product that still contains registry entries or queue managers, or both. In this situation, the new installation uses the old data directory location. For more information, see Program and data directory locations.

IBM i On IBM i, IBM MQ can only be installed in the default location. For more information about the directory structure of IBM i, see Directory structure on IBM i

UNIX Linux On UNIX and Linux systems, working data is stored in `/var/mqm`, but you cannot change this location. For more information about the directory structure of UNIX and Linux systems, see Directory structure on UNIX and Linux systems.

Custom location installation

For an installation into a custom location, the path specified must either be an empty directory, or a path that does not exist. The length of the path is limited to 256 bytes. Permissions on the path must be such that the user mqm and users in the mqm group can access the directories.

- UNIX Linux On UNIX and Linux systems, the path must not contain spaces.
- AIX On AIX, the product is installed into a User Specified Installation Location (USIL), which can be either an existing USIL or a new USIL that is automatically created by the installation process. If a custom location is specified, the product location is the path specified during installation, plus `/usr/mqm`.

  For example, the path specified is `/usr/custom_location`. The *`MQ_INSTALLATION_PATH`* is `/usr/custom_location/usr/mqm`.

  Access permissions for the USIL directory should be set to rwx for user and r-x for group and others (755).
- HP-UX Linux Solaris Windows On Windows, Linux, HP-UX, and Solaris, the product location is the same path as specified during installation.

  For example, on Linux, the path specified is `/opt/custom_location`. The *`MQ_INSTALLATION_PATH`* is `/opt/custom_location`.
- Linux Solaris HP-UX On Linux, Solaris, and HP-UX, you can install IBM MQ into a non empty MQ_INSTALLATION_PATH directory.

  Linux Solaris On Linux and Solaris you do this by setting the environment variable AMQ_OVERRIDE_EMPTY_INSTALL_PATH to 1 before starting the installation.

  HP-UX On HP-UX you need to create the file `/tmp/AMQ_OVERRIDE_EMPTY_INSTALL_PATH` before starting the installation.

  Note, that a non empty directory in this context, indicates a directory which contains system files and directories.

For each installation, all of the IBM MQ components that you require must be installed in the same location.

For more information about how to install to a custom location, see the installation topics for the appropriate platform.

Additional location restrictions

New IBM MQ installations should not be located in the following paths:
- In a path that is a subdirectory of another existing installation.
- In a path that is part of the direct path to an existing installation.
- In a path that is a subdirectory of the default location, for example:
  - AIX /usr/mqm on AIX
  - HP-UX Linux Solaris /opt/mqm on Linux, Solaris and HP-UX platforms
- In a directory or subdirectory that is, or might later be used by another product, for example, an IBM Db2 installation, or operating system component.

HP-UX Linux Solaris An installation should not be located in /opt/mqm/v80, /opt/mqm/v75, /opt/mqm/inst2/mq71, or other directory located under /opt/mqm on Linux, Solaris and HP-UX platforms.

If IBM MQ is installed in /opt/IBM/MQ/installations/1, you cannot install in /opt/IBM/MQ/installations/1/a. Additionally, you should not install a new installation to /opt/IBM/MQ. However, you can install a new installation in /opt/IBM/MQ/installations/2 or /opt/IBM/MQnew because neither of these is a part of the direct path /opt/IBM/MQ/installations/1.

You must not install to any directory located under /opt/IBM/db2.

The reason an installation should not be located in a path that is a subdirectory of the default location is to avoid the risk if you later decide to install IBM MQ into the default location, and cannot then do so. If you do subsequently install into the default location, because IBM MQ has full access rights over the installation directory, existing files might be replaced or deleted. Scripts that you might subsequently run to uninstall IBM MQ might remove the installation directory at the end of the script.

Related concepts:

"Planning considerations for installation on Multiplatforms" on page 196
Before you install IBM MQ, you must choose which components to install and where to install them. You must also make some platform-specific choices.

"Installation name on UNIX, Linux, and Windows" on page 197
Each installation of IBM MQ on UNIX, Linux, and Windows, has a unique identifier known as an installation name. The installation name is used to associate things such as queue managers and configuration files with an installation.

"Primary installation on UNIX, Linux, and Windows" on page 202
On systems that support multiple installations of IBM MQ (UNIX, Linux, and Windows), the primary installation is the one to which IBM MQ system-wide locations refer. Having a primary installation is optional, but convenient.

"IBM MQ components and features" on page 192
You can select the components or features that you require when you install IBM MQ.

Multiple installations on UNIX, Linux, and Windows

ULW

On UNIX, Linux, and Windows, it is possible to have more than one copy of IBM MQ on a system.

You can choose where each copy of IBM MQ is installed, but each copy must be in a separate installation location. A maximum of 128 installations of IBM MQ can exist on a single machine at a time. One installation can be an installation of IBM WebSphere MQ Version 7.0.1, Fix Pack 6, or later. You have a choice:

- Keep the simplicity of maintaining and managing a single installation of IBM MQ on a machine.
- Take advantage of the flexibility that is offered by enabling multiple IBM MQ installations.

Decisions to make before installing

Before you install multiple copies of IBM MQ, you must make several decisions:

Will you have a copy of IBM WebSphere MQ Version 7.0.1 on the system?
When IBM WebSphere MQ Version 7.0.1, Fix Pack 6, or later, is installed on the system, there are a number of restrictions to consider:

- Linux UNIX On UNIX and Linux systems, IBM WebSphere MQ Version 7.0.1 must be installed in the default location.

- IBM WebSphere MQ Version 7.0.1 must be the first installation on a system. You cannot install IBM WebSphere MQ Version 7.0.1 after installing Version 7.1, or later. If you uninstall version Version 7.0.1, it cannot be reinstalled while a later version of IBM MQ is installed.
- IBM WebSphere MQ Version 7.0.1 is automatically the primary installation. You cannot select another installation as the primary installation while IBM WebSphere MQ Version 7.0.1 is installed.

Where will you install each copy of IBM MQ ?
You can choose the installation location for your installations at Version 7.1, or later. For more information, see "Installation location on Multiplatforms" on page 198.

Do you need a primary installation?
A primary installation is an installation to which system-wide locations refer.

For more information, see "Primary installation on UNIX, Linux, and Windows" on page 202.

How will your applications connect?
You need to consider how your applications locate the appropriate IBM MQ libraries. For more information, see Connecting applications in a multiple installation environment, and Connecting .NET applications in a multiple installation environment.

Do your existing exits need changing?
If IBM MQ is not installed in the default location, your exits need to be updated. For more information, see Writing exits and installable services on UNIX, Linux, and Windows.

Which queue manager will be associated with which installation?
Each queue manager is associated with a particular installation. 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. For more information, see Associating a queue manager with an installation.

How will you set up your environment to work with each installation?
With multiple installations on a system, you need to consider how you will work with particular installations, and how you will issue commands from that installation. You can either specify the full path to the command, or you can use the **`setmqenv`** or **`crtmqenv`** command to set environment variables. Setting the environment variables allows you to omit the path to the commands for that installation. For more information, see setmqenv, and crtmqenv.

When you have answered these questions, you can install IBM MQ after you have read "IBM MQ installation overview" on page 192.

If you have existing installations of IBM MQ and you want to use the multiple installation capability to migrate from one version of IBM MQ to another version, see one of the following platform-specific topics:

- ULW Multi-installation queue manager coexistence on UNIX, Linux, and Windows
- NSS Client "Planning your client installation on HP Integrity NonStop Server" on page 252

The IBM message service client for .NET support pack and multiple installations

For multiple version support, on IBM WebSphere MQ Version 7.1 or later, the *Java and .NET Messaging and Web Services* feature must be installed with the IBM MQ product. For more information about installing the .NET feature, see Installing IBM MQ classes for .NET.

Related tasks:

"Choosing MSI Instance IDs for multiple server installations" on page 461
For multiple silent installations, for each version that is installed you must find an MSI instance ID that is available to use for that installation.

"Choosing MSI Instance IDs for multiple client installations" on page 486
For multiple silent installations, for each version that is installed you must find an MSI instance ID that is available to use for that installation.

Related information:

Configuring multiple installations

Finding installations of IBM MQ on a system

UNIX, Linux, and Windows: Side-by-side migration from Version 7.0.1, or later, to the latest version

UNIX, Linux, and Windows: Multi-stage migration from Version 7.0.1, or later, to the latest version

Primary installation on UNIX, Linux, and Windows

ULW

On systems that support multiple installations of IBM MQ (UNIX, Linux, and Windows), the primary installation is the one to which IBM MQ system-wide locations refer. Having a primary installation is optional, but convenient.

Before IBM WebSphere MQ Version 7.1, only one instance of the product could be installed at any time. On Windows systems, several global environment variables were set to point to that installation. On UNIX and Linux systems, symbolic links were added to `/usr/lib`, `/usr/bin`, and `/usr/include`, also pointing at that single installation.

From Version 7.1, you can install multiple versions of IBM MQ on UNIX, Linux, and Windows. It is possible to have more than one installation of IBM MQ on one of these systems at any time and, optionally, to configure one of these installations as the primary installation. Environment variables and symbolic links pointing to a single installation are less meaningful when multiple versions exist. However, some functions require these system-wide locations to work. For example, custom user scripts for administering IBM MQ, and third party products. These functions work only on the primary installation.

Linux UNIX On UNIX and Linux systems, if you set an installation as the primary installation, symbolic links to the external libraries and control commands of that installation are added into `/usr/lib`, and `/usr/bin`. If you do not have a primary installation, the symbolic links are not created. For a list of the symbolic links that are made to the primary installation, see "External library and control command links to primary installation on UNIX and Linux" on page 206.

Windows On Windows systems, the global environmental variables point to the directories into which the primary installation was installed. These environment variables are used to locate IBM MQ libraries, control commands, and header files. Additionally, on Windows systems, some features of the operating system require the central registration of interface libraries that are then loaded into a single process. With multiple versions of IBM MQ, there would be conflicting sets of IBM MQ libraries. The features would try to load these conflicting sets of libraries into a single process. Therefore, such features can be used only with the primary installation. For details about some of the features that are limited to use with the primary installation, see "Features that can be used only with the primary installation on Windows" on page 209.

If you have an installation of IBM WebSphere MQ Version 7.0.1 on the system, this installation is automatically the primary installation. The primary installation cannot be changed while Version 7.0.1 is installed. If all the installations on the system are at Version 7.1, or later, you can choose whether to have a primary installation. Consider the options in Table 30 on page 203.

Table 30. Primary installation options

Options	Valid installation configurations		More information
	Primary	Non-primary	
Single installation of Version 7.1, or later.	Version 7.1, or later.	None	If you want to continue working with a single installation in the same way as previous releases, configure your installation as the primary installation. For information about this option, see Single installation of IBM WebSphere MQ Version 7.1, or later, configured as the primary installation
	None	Version 7.1, or later.	If you want to continue working with a single installation, but do not want symbolic links or global environment variables created for you, configure your installation as non-primary. For information about the implications of this option, see Single installation of IBM WebSphere MQ Version 7.1, or later, configured as non-primary
Multiple installations: Version 7.0.1 and Version 7.1, or later.	Version 7.0.1	Version 7.1, or later.	If you want to have multiple installations of IBM MQ, with one at Version 7.0.1, the Version 7.0.1 installation is automatically the primary installation. While IBM WebSphere MQ Version 7.0.1 is installed, you cannot change which installation is the primary installation. For information about this option and its implications, see Multiple installations of IBM MQ, one at Version 7.0.1
Multiple installations: Version 7.1, or later.	Version 7.1, or later.	Version 7.1, or later.	If you want to have multiple installations of IBM MQ at Version 7.1 or greater, you can choose whether to make one of the installations primary. For information about this option, see Multiple installations of IBM WebSphere MQ Version 7.1, or later
	None	Version 7.1, or later.	

Related concepts:

"Installation location on Multiplatforms" on page 198
You can install IBM MQ into the default location. Alternatively, you can install into a custom location during the installation process. The location where IBM MQ is installed is known as the *MQ_INSTALLATION_PATH*.

"Planning considerations for installation on Multiplatforms" on page 196
Before you install IBM MQ, you must choose which components to install and where to install them. You must also make some platform-specific choices.

"Installation name on UNIX, Linux, and Windows" on page 197
Each installation of IBM MQ on UNIX, Linux, and Windows, has a unique identifier known as an installation name. The installation name is used to associate things such as queue managers and configuration files with an installation.

Related information:

Single installation of IBM WebSphere MQ Version 7.1, or later, configured as the primary installation
Marking an IBM MQ installation as primary adds symbolic links, or global environment variables to the system so that the IBM MQ commands and libraries used by applications are automatically available with minimum system setup required.

Single installation of IBM WebSphere MQ Version 7.1, or later, configured as non-primary
If you install IBM WebSphere MQ Version 7.1, or later, as non-primary you might have to configure a library path for applications to load IBM MQ libraries. On Windows, some product capabilities are available only when IBM MQ is configured as primary.

Multiple installations of IBM WebSphere MQ Version 7.1, or later
You can choose to have one of the IBM WebSphere MQ Version 7.1 installations configured as the primary installation. Your choice depends on how applications locate libraries.

Multiple installations of IBM MQ, one at Version 7.0.1
IBM WebSphere MQ Version 7.1, or later, can coexist with IBM WebSphere MQ Version 7.0.1 with some limitations.

Changing the primary installation

Single installation of Version 7.1, or later, configured as the primary installation: ULW

Marking an IBM MQ installation as primary adds symbolic links, or global environment variables to the system so that the IBM MQ commands and libraries used by applications are automatically available with minimum system setup required.

You decide where to install IBM MQ.

Where possible, configure applications and scripts to use the system search path to find the IBM MQ control commands or IBM MQ libraries. This configuration of applications and scripts provides maximum flexibility for undertaking future tasks such as migrating to the next release of IBM MQ, or installing a second installation. For more information about options for connecting your applications, see Connecting applications in a multiple installation environment.

Windows On Windows, the first installation is automatically configured as the primary installation.

Linux UNIX On UNIX and Linux platforms, the first installation onto a system must be manually configured to be the primary installation.

Set the primary installation using the **setmqinst** command. For more information, see Uninstalling, upgrading, and maintaining the primary installation.
Related information:
Changing the primary installation
Choosing an installation location
Planning your installation
Choosing an installation name

Single installation of Version 7.1, or later, configured as non-primary: ULW

If you install IBM WebSphere MQ Version 7.1, or later, as non-primary you might have to configure a library path for applications to load IBM MQ libraries. On Windows, some product capabilities are available only when IBM MQ is configured as primary.

Linux UNIX
UNIX and Linux systems

The implications of running a non-primary installation on UNIX and Linux are:
- Applications that locate their IBM MQ libraries using an embedded library path, for example, RPATH, cannot find those libraries if the following conditions are true:
  - IBM MQ is installed in a different directory from the directory specified in the RPATH
  - There are no symbolic links in /usr
- Where applications locate their libraries using an external library path, for example, LD_LIBRARY_PATH, you must configure the external library path to include the *MQ_INSTALLATION_PATH*/lib or *MQ_INSTALLATION_PATH*/lib64 directory. The **setmqenv** and **crtmqenv** commands can configure a number of environment variables in the current shell, including the external library path.
- Most IBM MQ processes run as setuid/setgid. As a result, when loading user exits they ignore the external library path. User exits that reference IBM MQ libraries can find those libraries only if they are found in the library path embedded within them. They would be resolved if there were a symbolic link in /usr. User exits that are intended to be run on IBM WebSphere MQ Version 7.1, or later can now be

built so that they do not refer to IBM MQ libraries at all. Instead they rely on IBM MQ to pass in function pointers to the IBM MQ functions that the exit can then use. For more information, see Writing exits and installable services on UNIX, Linux, and Windows.

For more information about options for connecting your applications, see Connecting applications in a multiple installation environment.

On UNIX and Linux platforms, the first installation onto a system is not automatically configured as the primary installation. However, a single symbolic link is included in `/usr/bin` to locate the **`dspmqver`** command. If you do not want any symbolic links, you must remove this link using the following command:

```
setmqinst -x -p MQ_INSTALLATION_PATH
```

Windows

Windows systems

The implications of running a non-primary installation on Windows are:
- Applications normally find their libraries using the external library path, `PATH`. There is no concept of an embedded library path or explicit library location. If the installation is non-primary, the global `PATH` environment variable does not contain the IBM MQ installation directory. For applications to find IBM MQ libraries, update the `PATH` environment variable to reference the IBM MQ installation directory. The **`setmqenv`** and **`crtmqenv`** commands can configure a number of environment variables in the current shell, including the external library path.
- Some product capabilities are available only when an installation is configured as the primary installation; see Features that can be used only with the primary installation on Windows.

By default, on Windows, the first installation is automatically configured as primary. You must manually deselect it as the primary installation.

Related information:

Changing the primary installation

Choosing an installation location

Planning your installation

setmqenv

crtmqenv

Choosing an installation name

Multiple installations of Version 7.1, or later: ULW

You can choose to have one of the IBM WebSphere MQ Version 7.1 installations configured as the primary installation. Your choice depends on how applications locate libraries.

The IBM MQ libraries, such as mqm, which ship with IBM WebSphere MQ Version 7.1 automatically use libraries of the level required by the queue manager to which they are connecting. This means that provided an application locates its IBM MQ libraries from a IBM WebSphere MQ Version 7.1 installation, it can connect to any queue manager on that system. Having one IBM WebSphere MQ Version 7.1 installation configured as primary ensures that if the application finds its IBM MQ interface library, the application can connect to any queue manager.

For more information about connecting applications in a multiple installation environment, see Connecting applications in a multiple installation environment.

The primary installation is not automatically changed when you uninstall the primary installation. If you want another installation to be the primary installation, you must manually set the primary installation using the **setmqinst** command. For more information, see Uninstalling, upgrading, and maintaining the primary installation.

Related information:

Changing the primary installation

Choosing an installation location

Multiple installations

Planning your installation

Choosing an installation name

Multiple installations, one at Version 7.0.1: ULW

IBM WebSphere MQ Version 7.1, or later, can coexist with IBM WebSphere MQ Version 7.0.1 with some limitations.

- On UNIX and Linux systems, Version 7.0.1 can be installed only in a fixed, default location, so you cannot install Version 7.1, or later, in that default location.
- IBM WebSphere MQ Version 7.0.1 is automatically configured as the primary installation. On UNIX and Linux systems, symbolic links are automatically created to the appropriate IBM MQ directories. On Windows, everything that the product provided is registered globally. IBM WebSphere MQ Version 7.0.1 must be installed in this way to work. So where IBM WebSphere MQ Version 7.0.1 is installed, a IBM WebSphere MQ Version 7.1, or later, installation cannot be made primary.

The libraries from IBM WebSphere MQ Version 7.1, or later, can work with any queue manager running under IBM WebSphere MQ Version 7.0.1, or later. If an application needs to connect to queue managers running under Version 7.0.1 as well as later versions, it can continue to operate normally if the following conditions are true:

- It locates IBM WebSphere MQ Version 7.1, or later, libraries at run time.
- It uses only functions available in Version 7.0.1.

For more information about connecting applications in a multiple installation environment, see Connecting applications in a multiple installation environment.

The primary installation is not automatically changed when you uninstall IBM WebSphere MQ Version 7.0.1. If you want another installation to be the primary installation, you must manually set the primary installation using the **setmqinst** command. For more information, see Uninstalling, upgrading, and maintaining the primary installation.

Related information:

Choosing an installation location

Planning your installation

Multiple installations

Choosing an installation name

External library and control command links to primary installation on UNIX and Linux: Linux UNIX

On UNIX and Linux platforms the primary installation is the one to which links from the `/usr` file system are made. However, only a subset of those links created with previous releases are now made.

No links are created from `/usr/include` to any installation and only links to external libraries and documented control commands are made from `/usr/lib`, and where appropriate, `/usr/lib64` (external libraries) and `/usr/bin` (control commands).

In order to run these commands you must complete the following steps:

1. provide a full path to the command in an available IBM MQ installation,
2. use the `setmqenv` script to update your shell environment,
3. manually add the bin directory from an IBM MQ installation directory to your PATH,
4. run the **`setmqinst`** command as root to make one of your existing IBM MQ installations the primary installation.

External libraries

Links are made to the following external libraries, both 32-bit and 64-bit:
- libmqm
- libmqm_r
- libmqmxa
- libmqmxa_r
- libmqmax
- libmqmax_r
- libmqmcb
- libmqmcb_r
- libmqic
- libmqic_r
- libmqcxa
- libmqcxa_r
- libmqicb
- libmqicb_r
- libimqb23ia
- libimqb23ia_r
- libimqc23ia
- libimqc23ia_r
- libimqs23ia
- libimqs23ia_r
- libmqmzf
- libmqmzf_r

The following 64-bit only libraries are also linked to:
- libmqmxa64
- libmqmxa64_r
- libmqcxa64
- libmqcxa64_r

Control commands

The following control commands are linked to from `/usr/bin`:
- addmqinf
- amqcrs6a
- amqcrsta
- amqmfsck
- crtmqinst
- dltmqinst

- dspmqinst
- setmqinst
- crtmqcvx
- crtmqm
- dltmqm
- dmpmqaut
- dmpmqlog
- dspmq
- dspmqaut
- dspmqcsv
- dspmqfls
- dspmqinf
- dspmqrte
- dspmqtrc
- dspmqtrn
- dspmqver
- endmqcsv
- endmqlsr
- endmqm
- endmqtrc
- rcdmqimg
- rcrmqobj
- rmvmqinf
- rsvmqtrn
- runmqchi
- runmqchl
- runmqckm
- runmqdlq
- runmqlsr
- runmqsc
- runmqtmc
- runmqtrm
- setmqaut
- setmqenv
- setmqm
- setmqprd
- strmqcsv
- strmqikm
- strmqm
- strmqtrc

Related concepts:

"Primary installation on UNIX, Linux, and Windows" on page 202
On systems that support multiple installations of IBM MQ (UNIX, Linux, and Windows), the primary installation is the one to which IBM MQ system-wide locations refer. Having a primary installation is optional, but convenient.

"Features that can be used only with the primary installation on Windows"
Some Windows operating-system features can be used only with the primary installation. This restriction is due to the central registration of interface libraries, which might conflict as a result of multiple versions of IBM MQ being installed.

Features that can be used only with the primary installation on Windows: Windows

Some Windows operating-system features can be used only with the primary installation. This restriction is due to the central registration of interface libraries, which might conflict as a result of multiple versions of IBM MQ being installed.

The .NET monitor

The IBM MQ .NET monitor can run in two different modes: transactional and non-transactional. The transactional mode uses MSDTC transaction coordination and requires that the .NET monitor is registered with COM+. The .NET monitor from the primary installation is the only .NET monitor that is registered with COM+.

Any attempt to run the .NET monitor in transactional mode with a non-primary installation results in the failure of the .NET monitor to enlist with MSDTC. The .NET monitor receives an `MQRC_INSTALLATION_MISMATCH` error, which in turn results in an AMQ8377 error message on the console.

COM/ActiveX interface classes

The COM/ActiveX interface classes are registered only for the primary installation. If there is an installation of IBM WebSphere MQ Version 7.0.1 on the system, the COM/ActiveX interface classes registered are not capable of connecting to queue managers running under other installations. If the primary installation is an installation of IBM WebSphere MQ Version 7.1 or later, the interface classes can connect to queue managers associated with any installation. Server COM/ActiveX applications are limited by this restriction, but client applications can connect to any queue manager.

Any attempt to start a COM/ActiveX application that uses libraries from installations other than the primary installation results in failure with an `MQRC_Q_MGR_NOT_AVAILABLE` error.

Related concepts:

"Primary installation on UNIX, Linux, and Windows" on page 202
On systems that support multiple installations of IBM MQ (UNIX, Linux, and Windows), the primary installation is the one to which IBM MQ system-wide locations refer. Having a primary installation is optional, but convenient.

"External library and control command links to primary installation on UNIX and Linux" on page 206
On UNIX and Linux platforms the primary installation is the one to which links from the `/usr` file system are made. However, only a subset of those links created with previous releases are now made.

Uninstalling, upgrading, and maintaining the primary installation: ULW

On all platforms, if you uninstall the primary installation, it ceases to be the primary installation. You must run the **`setmqinst`** command to select a new primary installation. On Windows, if you update the primary installation, it continues to be the primary installation. If you apply a fix pack to the primary installation, it continues to be the primary installation.

Be cautious about the effect uninstalling or upgrading the primary installation has on applications. Applications might be using the linkage library of the primary installation to switch to the linkage library of another installation. If such an application is running, you might not be able to uninstall the primary installation. The operating system might have locked the link library of the primary installation on behalf of the application. If the primary installation has been uninstalled, an application that loads the IBM MQ libraries it requires by linking to the primary installation is not able to start.

The solution is to switch the primary installation to another installation before uninstalling. Stop, and restart applications that are linked through the previous primary installation before uninstalling it.

Windows

If you update the primary installation, it stops being the primary installation at the beginning of the update procedure. If, by the end of the update procedure, you have not made another installation primary, the upgraded installation is made primary again.

Maintenance

If you apply a fix pack to the primary installation, it stops being the primary installation at the beginning of the maintenance procedure. If, by the end of the maintenance procedure, you have not made another installation primary, the upgraded installation is made primary again.

Related concepts:

"Primary installation on UNIX, Linux, and Windows" on page 202
On systems that support multiple installations of IBM MQ (UNIX, Linux, and Windows), the primary installation is the one to which IBM MQ system-wide locations refer. Having a primary installation is optional, but convenient.

Related information:

Changing the primary installation

IBM MQ maintenance tasks

Server-to-server links on UNIX, Linux, and Windows

For verifying a server-to-server verification, the communication links between the two systems must be checked. Before you can do the verification, you must ensure that the communications protocol is installed and configured on both systems.

The examples used in the verification tasks listed in this topic for UNIX, Linux, and Windows use TCP/IP.

The various communication protocols used by the supported platforms are as follows.

UNIX
: IBM MQ supports both TCP and SNA. If you do not use TCP, see Setting up communication on UNIX and Linux systems.

Linux
: IBM MQ for Linux supports TCP on all Linux platforms. On x86 platforms and Power platforms, SNA is also supported. If you want to use the SNA LU6.2 support on these platforms, you need the IBM Communications Server for Linux Version 6.2. The Communications Server is available as a PRPQ product from IBM. For more details, see Communications Server.

  If you do not use TCP, see Setting up communication on UNIX and Linux systems.

Windows **Windows**

IBM MQ for Windows supports TCP, SNA, NetBios, and SPX. If you do not use TCP, see Setting up communication for Windows.

Related tasks:

"Verifying an IBM MQ installation on AIX" on page 233
The topics in this section provide instructions on how to verify a server or a client installation of IBM MQ on AIX systems.

"Verifying an IBM MQ installation on HP-UX" on page 282
The topics in this section provide instructions on how to verify a server or a client installation of IBM MQ on HP-UX systems.

"Verifying an IBM MQ installation on Linux" on page 377
The topics in this section provide instructions on how to verify a server or a client installation of IBM MQ on Linux systems.

"Verifying an IBM MQ installation on Solaris" on page 420
The topics in this section provide instructions on how to verify a server or a client installation of IBM MQ on Solaris systems.

"Verifying an IBM MQ installation on Windows" on page 496
The topics in this section provide instructions on how to verify a server or a client installation of IBM MQ on Windows systems.

Installation with a download image

Multi

You can perform an installation of IBM MQ from an installation image downloaded from Passport Advantage®.

Before you start installing, consider how you want to use IBM MQ and review the general Planning section.

Go to the Passport Advantage and Passport Advantage Express® web site for further information on how you:

- Acquire new IBM software licenses.
- Renew Software Subscription and Support and Fixed Term Licenses.
- Buy and renew technical support for some Selected Open Source and other non-warranted applications.
- Subscribe to IBM SaaS offerings and acquire IBM Appliances.

Passport Advantage is designed for larger enterprises and enterprises with multiple sites.

Passport Advantage Express is designed for smaller enterprises and single-site enterprises.

Related information:

Downloading IBM MQ Version 9.0

Redistributable clients

Linux Windows

The IBM MQ redistributable client is a collection of runtime files that are provided in a `.zip` or `.tar` file that can be redistributed to third parties under redistributable license terms, which provides a simple way of distributing applications and the runtime files that they require in a single package.

What are the IBM MQ redistributable clients?

From IBM MQ Version 8.0.0, Fix Pack 4, native redistributable client runtime libraries are provided for Linux x86-64 and Windows 64-bit platforms to make it simple to distribute both applications and the required IBM MQ runtime libraries. A third package, which is not platform-specific, contains the runtime files that are required for the Java/JMS applications, including the IBM MQ resource adapter for JMS applications that are running under an application server.

The redistributable client that is supplied with IBM MQ is also a non-installed and relocatable image. Maintenance of a redistributable, non-installed image, is achieved through replacement; that is, you download newer versions of the runtime components when they are shipped.

A *redistributable* client implies distributing the required run time with an application both inside and outside of your environment.

A *relocatable* client implies putting the files somewhere else other than a fixed default location. For example, instead of installing into `/opt/` installing into `/usr/local`.

A *non-installed* client implies that you are not required to lay down client files, and that these files can be copied as required.

The IBM IPLA license agreement is extended for IBM MQ to enable you to download a number of additional runtime files from Fix Central.

Supported languages

You can use the files that are contained in the redistributable images to run the following client applications:
- C
- C++
- COBOL
- Java
- Java JMS
- Both fully managed and unmanaged .NET

Limitations

GSKit objects
No new GSKit objects are being shipped. Only the runtime files are shipped, both in a regular installation and with the redistributable client.

IBM JREs
No IBM JREs are being provided with the redistributable client.

If you want to run Java/JMS applications, you must provide your own runtime environment. Your JRE, that applications run under, must meet the current SOE requirements and are bound by any restrictions or limitations that apply.

Developing applications
All other files that support the development and distribution of applications (including copybooks, header files, and sample source code) are not included in the redistributable client and are not licensed for redistribution.

If you need to develop IBM MQ applications, you still need to perform a traditional installation so that you obtain the SDK files that are required to build client applications.

Windows **Windows C runtime libraries**
You might have these libraries on your machine already, but if you do not, you need to download and install the following Microsoft C/C++ runtime libraries:

- Microsoft Visual C++ Redistributable 2008
- Microsoft Visual C++ Redistributable 2012

The download links for the redistributable downloads for each of these libraries can be found at The latest supported Visual C++ downloads .

For more information about installing redistributable clients, see:

- Linux "Redistributable clients on Linux" on page 374
- Windows "Redistributable clients on Windows" on page 493

Related concepts:

"Planning considerations for installation on Multiplatforms" on page 196
Before you install IBM MQ, you must choose which components to install and where to install them. You must also make some platform-specific choices.

"Installation location on Multiplatforms" on page 198
You can install IBM MQ into the default location. Alternatively, you can install into a custom location during the installation process. The location where IBM MQ is installed is known as the *MQ_INSTALLATION_PATH*.

Windows ".NET application runtime - Windows only" on page 494
Considerations when using the .NET application.

Related information:

Configuring the Redistributable Managed File Transfer Agent

Installation considerations for redistributable clients

Linux Windows

The Linux x86-64 image is shipped in a `LinuxX64.tar.gz` file, and the Windows 64-bit image is shipped in a `Win64.zip` file.

File names

The archive or `.zip` file names describe the file contents and equivalent maintenance levels.

CD For example, for Continuous Delivery, in IBM MQ Version 9.0.4 the client images are available under the following file names:

Linux **Linux x86-64**
`9.0.4.0-WS-MQC-Redist-LinuxX64.tar.gz`

Windows **Windows**
`9.0.4.0-WS-MQC-Redist-Win64.zip`

Linux Windows z/OS **Java**
`9.0.4.0-WS-MQC-Redist-Java.zip`

LTS For Long Term Support, in IBM MQ Version 9.0.0, Fix Pack 2 the client images are available under the following file names:

Linux **Linux x86-64**
`9.0.0.2-WS-MQC-Redist-LinuxX64.tar.gz`

Windows **Windows**
`9.0.0.2-WS-MQC-Redist-Win64.zip`

Java `9.0.0.2-WS-MQC-Redist-Java.zip`

Choosing the runtime files to distribute with an application

A script file named **genmqpkg** is provided by the redistributable client under the `bin` directory.

You can use the **genmqpkg** script to generate a smaller subset of files that are tailored to the needs of the application, for which the files are intended to be distributed.

You are asked a series of interactive `Yes` or `No` questions to determine the runtime requirements for an IBM MQ application.

Finally, **genmqpkg** asks you to supply a new target directory, where the script duplicates the required directories and files.

Important: IBM support is only able to provide assistance with the full, unmodified set of files contained within the redistributable client packages.

Other considerations

The default data path of a non-installed client is:

Linux **Linux x86-64**
`$HOME/IBM/MQ/data`

Windows **Windows**
`%HOMEDRIVE%\%HOMEPATH%\IBM\MQ\data`

Important: A redistributable client runtime co-exists with a full IBM MQ client or server installation, provided that they are installed in different locations. However, unpacking a redistributable image into the same location as a full IBM MQ installation is not supported.

On Linux the `ccsid.tbl` used to define the supported CCSID conversions is traditionally expected to be found in the `UserData` directory structure, along with error logs, trace files, and so on. The `UserData` directory structure is populated by unpacking the redistributable client, and so, if the file is not found in its usual location, the redistributable client falls back to locate the file in the `/lib` subdirectory of the installation.

Classpath changes

The classpath used by **`dspmqver`**, **`setmqenv`**, and **`crtmqenv`** commands adds the `com.ibm.mq.allclient.jar` to the environment, immediately following the `com.ibm.mq.jar` and `com.ibm.mqjms.jar`.

An example of **`dspmqver`** output from the redistributable client on Linux:

```
Name:        IBM MQ
Version:     8.0.0.4
Level:       p800-804-L150909
BuildType:   IKAP - (Production)
Platform:    IBM MQ for Linux (x86-64 platform)
Mode:        64-bit
O/S:         Linux 2.6.32.59-0.7-default
InstName:    MQNI08000004
InstDesc:    IBM MQ V8.0.0.4 (Redistributable)
Primary:     No
InstPath:    /Development/johndoe/unzip/unpack
DataPath:    /u/johndoe/IBM/MQ/data
MaxCmdLevel: 802
```

An example of **`dspmqver`** output from the redistributable client on Windows:

```
Name:        IBM MQ
Version:     8.0.0.4
Level:       p800-804-L150909
BuildType:   IKAP - (Production)
Platform:    IBM MQ for Windows (x64 platform)
Mode:        64-bit
O/S:         Windows 7 Professional x64 Edition, Build 7601: SP1
InstName:    MQNI08000004
InstDesc:    IBM MQ V8.0.0.4 (Redistributable)
Primary:     No
InstPath:    C:\Users\johndoe\Desktop\Redist
DataPath:    C:\Users\johndoe\IBM\MQ\data
MaxCmdLevel: 802
```

Related concepts:

"Redistributable clients" on page 212
The IBM MQ redistributable client is a collection of runtime files that are provided in a `.zip` or `.tar` file that can be redistributed to third parties under redistributable license terms, which provides a simple way of distributing applications and the runtime files that they require in a single package.

".NET application runtime - Windows only" on page 494
Considerations when using the .NET application.

Installing and uninstalling IBM MQ on AIX

AIX

Installation tasks that are associated with installing IBM MQ on AIX systems are grouped in this section.

About this task

To prepare for installation and to install the IBM MQ components, complete the following tasks.

For information about how to uninstall IBM MQ, see "Uninstalling or modifying IBM MQ on AIX" on page 247.

If product fixes or updates are made available, see IBM MQ maintenance tasks for information about how to apply these changes.

Procedure

1. Check the system requirements. See "Checking requirements on AIX" on page 218.
2. Plan your installation.

- As part of the planning process, you must choose which components to install and where to install them. See "IBM MQ components for AIX."
- You must also make some platform-specific choices. See "Planning to install IBM MQ on AIX" on page 220.

3. Prepare your system for installation of IBM MQ. See "Preparing the system on AIX" on page 221.
4. Install IBM MQ server. See "Installing IBM MQ server on AIX" on page 225.
5. Optional: Install an IBM MQ client. See "Installing an IBM MQ client on AIX" on page 231.
6. Verify your installation. See "Verifying an IBM MQ installation on AIX" on page 233.

IBM MQ components for AIX

AIX

You can select the components that you require when you install IBM MQ.

Important: See IBM MQ license information for details of what each purchase of IBM MQ entitles you to install.

On AIX each component of IBM MQ is represented by a fileset. Table 31 shows the filesets that are available when installing an IBM MQ server or client on an AIX system:

Table 31. IBM MQ filesets for AIX systems

Component	Description	Server media	Client media	Fileset name
Runtime	Contains files that are common to both server and client installations. **Note:** This component must be installed.	✓	✓	mqm.base.runtime
Server	You can use the server to run queue managers on your system and connect to other systems over a network. Provides messaging and queuing services to applications, and support for IBM MQ client connections.	✓		mqm.server.rte
Standard Client	The IBM MQ MQI client is a small subset of IBM MQ, without a queue manager, that uses the queue manager and queues on other (server) systems. It can be used only when the system it is on is connected to another system that is running a full server version of IBM MQ. The client and the server can be on the same system if required.	✓	✓	mqm.client.rte
SDK	The SDK is required for compiling applications. It includes sample source files, and the bindings (files .H, .LIB, .DLL, and others), that you need to develop applications to run on IBM MQ.	✓	✓	mqm.base.sdk
Sample programs	The sample application programs are needed if you want to check your IBM MQ installation using the verification procedures.	✓	✓	mqm.base.samples
Java messaging	The files needed for messaging using Java (includes Java Message Service).	✓	✓	mqm.java.rte
Man pages	UNIX man pages, in U.S. English, for: control commands MQI calls MQSC commands	✓	✓	mqm.man.en_US.data

Table 31. IBM MQ filesets for AIX systems (continued)

Component	**Description**	**Server media**	**Client media**	**Fileset name**
Java JRE	A Java Runtime Environment that is used by those parts of IBM MQ that are written in Java.	✓	✓	mqm.jre.rte
Message Catalogs	For available languages, see the table of message catalogs that follows.	✓	✓	
IBM Global Security Kit	IBM Global Security Kit V8 Certificate and TLS, Base Runtime.	✓	✓	mqm.gskit.rte
Telemetry Service	MQ Telemetry supports the connection of Internet Of Things (IOT) devices (that is, remote sensors, actuators and telemetry devices) that use the IBM MQ Telemetry Transport (MQTT) protocol. The telemetry service, which is also know as the MQXR service, enables a queue manager to act as an MQTT server, and communicate with MQTT client apps. A set of MQTT client libraries is also available in the IBM Messaging Telemetry Clients SupportPac. These libraries help you write the MQTT client apps that IOT devices use to communicate with MQTT servers. See also "Installation considerations for MQ Telemetry" on page 538.	✓		mqm.xr.service
Managed File Transfer	MQ Managed File Transfer transfers files between systems in a managed and auditable way, regardless of file size or the operating systems used. For information about the function of each component, see Managed File Transfer product options.	✓		mqm.ft.agent mqm.ft.base mqm.ft.logger mqm.ft.service mqm.ft.tools
Advanced Message Security	Provides a high level of protection for sensitive data flowing through the IBM MQ network, while not impacting the end applications. You must install this component on all IBM MQ installations that host queues you want to protect. You must install the IBM Global Security Kit component on any IBM MQ installation that is used by a program that puts or gets messages to or from a protected queue, unless you are using only Java client connections. You must install the **Java JRE** component to install this component.	✓		mqm.ams.rte
V 9.0.0 AMQP Service	Install this component to make AMQP channels available. AMQP channels support MQ Light APIs. You can use AMQP channels to give AMQP applications access to the enterprise-level messaging facilities provided by IBM MQ.	✓		mqm.amqp.rte
V 9.0.4 REST API and Console	Adds HTTP based administration for IBM MQ through the REST API and IBM MQ Console.	✓		mqm.web.rte

Table 32. IBM MQ message catalogs for AIX systems

Message catalog language	Component name
Brazilian Portuguese	mqm.msg.pt_BR
Czech	mqm.msg.cs_CZ
French	mqm.msg.fr_FR
German	mqm.msg.de_DE
Hungarian	mqm.msg.hu_HU
Italian	mqm.msg.it_IT
Japanese	mqm.msg.ja_JP, mqm.msg.Ja_JP
Korean	mqm.msg.ko_KR
Polish	mqm.msg.pl_PL
Russian	mqm.msg.ru_RU
Spanish	mqm.msg.es_ES
Simplified Chinese	mqm.msg.zh_CN, mqm.msg.Zh.CN
Traditional Chinese	mqm.msg.zh_TW, mqm.msg.Zh_TW
U.S. English	mqm.msg.en_US

Related concepts:

"IBM MQ components and features" on page 192
You can select the components or features that you require when you install IBM MQ.

"Planning considerations for installation on Multiplatforms" on page 196
Before you install IBM MQ, you must choose which components to install and where to install them. You must also make some platform-specific choices.

Checking requirements on AIX

AIX

Before you install IBM MQ on AIX, you must check for the latest information and system requirements.

About this task

A summary of the tasks that you must complete to check system requirements is listed here with links to further information.

Procedure

1. Check that you have the latest information, including information on hardware and software requirements. See "Where to find product requirements and support information" on page 195.
2. Check that your systems meet the initial hardware and software requirements for AIX. See "Hardware and software requirements on AIX systems" on page 219.

   The supported hardware and software environments are occasionally updated. See System Requirements for IBM MQ for the latest information.
3. Check that your systems have sufficient disk space for the installation. See Disk space requirements.
4. Check that you have the correct licenses. See "License requirements" on page 194 and IBM MQ license information.

What to do next

When you have completed these tasks, you are ready to start preparing your system for installation. For the next steps in installing IBM MQ, see "Preparing the system on AIX" on page 221.

Related concepts:

"IBM MQ installation overview" on page 192
An overview of concepts and considerations for installing IBM MQ, with links to instructions on how to install, verify, and uninstall IBM MQ on each of the supported platforms.

Related information:

IBM MQ maintenance tasks

Hardware and software requirements on AIX systems

AIX

Before you install IBM MQ, check that your system meets the hardware and operating system software requirements for the particular components you intend to install.

For hardware and software requirements, see System Requirements for IBM MQ.

IBM MQ does not support host names that contain spaces. If you install IBM MQ on a system with a host name that contains spaces, you are unable to create any queue managers.

Java Message Service and SOAP transport

If you want to use Java Message Service and SOAP support, you need an IBM Java 7 SDK and Runtime Environment Version 7.0 or later.

V 9.0.0 Java 8 is bundled with IBM MQ Version 9.0 but client components are built with Java 7 compatibility flags on.

For development, a JDK is required, and a JRE is required for running. The JRE does not need to be the JRE installed with IBM MQ, but has to be one from the supported list.

For a list of supported JDKs, see System Requirements for IBM MQ.

For further information about SOAP with IBM MQ, see IBM MQ transport for SOAP.

You can check the version installed using the following command:

```
java -version
```

Transport Layer Security (TLS)

If you want to use the TLS support, you need the IBM Global Security Kit (GSKit) V8 package. This package is supplied with IBM MQ as one of the components available for installation.

Unicode support on AIX

If you need to convert data to and from Unicode on your system, you must install the following file sets:

```
bos.iconv.ucs.com   Unicode converters for AIX sets
bos.iconv.ucs.ebcdic Unicode converters for EBCDIC sets
bos.iconv.ucs.pc   Unicode converters for PC sets
```

Related concepts:

IBM i "Hardware and software requirements on IBM i systems" on page 299
Check that the server environment meets the prerequisites for installing IBM MQ for IBM i. Check the product readme files and install missing prerequisite software supplied on the server CD.

"Hardware and software requirements on Windows systems" on page 445
Check that the server environment meets the prerequisites for installing IBM MQ for Windows and install any prerequisite software that is missing from your system from the server DVD.

Related tasks:

"Checking requirements on Windows" on page 444
Before you install IBM MQ on Windows, you must check for the latest information and system requirements.

Planning to install IBM MQ on AIX

AIX

Before you install IBM MQ on AIX, you must choose which components to install and where to install them. You must also make some platform-specific choices.

About this task

The following steps provide links to additional information to help you with planning your installation of IBM MQ on AIX.

As part of your planning activities, make sure that you review the information on hardware and software requirements for the platform on which you are planning to install IBM MQ. For more information, see "Checking requirements on AIX" on page 218.

Procedure

- Decide which IBM MQ components and features to install. See "IBM MQ components and features" on page 192.

  Important: Ensure that your enterprise has the correct license, or licenses, for the components that you are going to install. For more information, see "License requirements" on page 194 and IBM MQ license information.
- Review the options for naming your installation. In some cases, you can choose an installation name to use instead of the default name. See "Installation name on UNIX, Linux, and Windows" on page 197.
- Review the options and restrictions for choosing an installation location for IBM MQ. For more information, see "Installation location on Multiplatforms" on page 198.
- If you plan to install multiple copies of IBM MQ, see "Multiple installations on UNIX, Linux, and Windows" on page 200.
- If you already have a primary installation, or plan to have one, see "Primary installation on UNIX, Linux, and Windows" on page 202.
- Make sure that the communications protocol needed for server-to-server verification is installed and configured on both systems that you plan to use. For more information, see "Server-to-server links on UNIX, Linux, and Windows" on page 210.

Preparing the system on AIX

AIX

On AIX systems, you might have to complete several tasks before you install IBM MQ. You might also want to complete other tasks, depending on your installation intentions.

About this task

The tasks that you perform to prepare your systems for installation are listed here. Complete the appropriate tasks for your platform before installing.

Procedure

1. Set up a user ID of the name mqm, with a primary group of mqm. See "Setting up the user and group on AIX."
2. Create file systems for both the product code and working data to be stored. See "Creating file systems on AIX" on page 223.
3. Configure any additional settings needed for your AIX system. See "Configuring and tuning the operating system on AIX" on page 224.

What to do next

When you have completed the tasks to prepare the system, you are ready to start installing IBM MQ. To install a server, see "Installing IBM MQ server on AIX" on page 225. To install a client, see "Installing an IBM MQ client on AIX" on page 231.

Related information:

Planning

Maintaining and migrating

IBM MQ maintenance tasks

Setting up the user and group on AIX

AIX

On AIX systems, IBM MQ requires a user ID of the name mqm, with a primary group of mqm. The mqm user ID owns the directories and files that contain the resources associated with the product.

Creating the user ID and group

Set the primary group of the mqm user to the group mqm.

If you are installing IBM MQ on multiple systems you might want to ensure each UID and GID of mqm has the same value on all systems. If you are planning to configure multi-instance queue managers, it is essential the UID and GID are the same from system to system. It is also important to have the same UID and GID values in virtualization scenarios.

AIX

You can use the System Management Interface Tool (smit), for which you require root authority.

1. To create the mqm group, display the required window using this sequence:

   ```
   Security & Users
   Groups
   Add a Group
   ```

   Set the group name field to mqm.
2. To create the user mqm, display the required window using this sequence:

```
Security & Users
Users
Add a User
```

Set the user name field to mqm.

3. To add a password to the new user ID, display the required window using this sequence:

   ```
   Security & Users
   Passwords
   Change a User's Password
   ```

   Set the password as required.

Adding existing user IDs to the group

If you want to run administration commands, for example **crtmqm** (create queue manager) or **strmqm** (start queue manager), your user ID must be a member of the mqm group. This user ID must not be longer than 12 characters.

Users do not need mqm group authority to run applications that use the queue manager; it is needed only for the administration commands.

AIX

You can use smit to add an existing user ID to the mqm group. Display the required menu using this sequence:

```
Security & Users
Users
Change / Show Characteristics of a User
```

Type the name of the user in the **User Name** field and press **Enter**. Add mqm to the **Group SET** field, which is a comma-separated list of the groups to which the user belongs. Users do not need to have their primary group set to mqm. If mqm is in their set of groups, they can use the administration commands.

Log files created by MQ Telemetry service

The **umask** setting of the user ID that creates a queue manager will determine the permissions of the Telemetry log files generated for that queue manager. Even though the ownership of the log files will be set to mqm.

Related concepts:

"Creating file systems on AIX" on page 223
Before installing IBM MQ, you might need to create file systems for both the product code and working data to be stored. There are minimum storage requirements for these file systems. The default installation directory for the product code can be changed at installation time, but the working data location cannot be changed.

"Configuring and tuning the operating system on HP-UX" on page 271
Before you install IBM MQ on an HP-UX system, you must check that the kernel is configured correctly.

"Configuring and tuning the operating system on Linux" on page 337
Use this topic when you are configuring IBM MQ on Linux systems.

Related tasks:

"Configuring and tuning the operating system on AIX" on page 224
When installing IBM MQ on AIX systems, there are some additional settings that must be configured.

Related information:

"Configuring and tuning the operating system on Solaris" on page 409
Configure Solaris systems with the resource limits required by IBM MQ.

Creating file systems on AIX

AIX

Before installing IBM MQ, you might need to create file systems for both the product code and working data to be stored. There are minimum storage requirements for these file systems. The default installation directory for the product code can be changed at installation time, but the working data location cannot be changed.

Determining the size of a server installations file system

To determine the size of the /var/mqm file system for a server installation, consider:
- The maximum number of messages in the system at one time.
- Contingency for message buildups, if there is a system problem.
- The average size of the message data, plus 500 bytes for the message header.
- The number of queues.
- The size of log files and error messages.
- The amount of trace that is written to the /var/mqm/trace directory.

Storage requirements for IBM MQ also depend on which components you install, and how much working space you need. For more details, see Disk space requirements.

Creating a file system for the working data

Before you install IBM MQ, create and mount a file system called /var/mqm which is owned by the user mqm in the group mqm; see "Setting up the user and group on AIX" on page 221. This file system is used by all installations of IBM MQ on a system. If possible, use a partition strategy with a separate volume for the IBM MQ data. This means that other system activity is not affected if a large amount of IBM MQ work builds up. Configure the directory permissions to permit the mqm user to have full control, for example, file mode 755. These permissions will then be updated during the IBM MQ installation to match the permissions required by the queue manager.

Creating separate file systems for errors and logs

You can also create separate file systems for your log data (/var/mqm/log) and error files (/var/mqm/errors). If possible, place these directories on different physical disks from the queue manager data (/var/mqm/qmgrs) and from each other.

If you create separate file systems the /var/mqm/errors directory can be NFS mounted. However, if you choose to NFS-mount /var/mqm/errors, the error logs might be lost if the network fails.

You can protect the stability of your queue manager by having separate file systems for:
- /var/mqm/errors
- /var/mqm/trace
- /var/mqm/qmgrs
- /var/mqm/log

In the case of /var/mqm/errors, it is rare that this directory receives large quantities of data. But it is sometimes seen, particularly if there is a severe system problem leading to IBM MQ writing a lot of diagnostic information in to .FDC files. In the case of /var/mqm/trace, files are only written here when you use **strmqtrc** to start tracing IBM MQ.

You can obtain better performance of normal IBM MQ operations (for example, syncpoints, MQPUT, MQGET of persistent messages) by placing the following on separate disks:
- /var/mqm/qmgrs

- /var/mqm/log

In the rare event that you need to trace an IBM MQ system for problem determination, you can reduce performance impact by placing the /var/mqm/trace file system on a separate disk.

If you are creating separate file systems, allow a minimum of 30 MB of storage for /var/mqm, 100 MB of storage for /var/mqm/log, and 10 MB of storage for /var/mqm/errors. The 100 MB minimum allowance of storage for /var/mqm/log is the absolute minimum required for a single queue manager and is not a recommended value. The size of a file system must be scaled according to the number of queue managers that you intend to use, the number of pages per log file, and the number of log files per queue manager.

If you want to use individual queues that hold more than 2 GB of data, you must enable /var/mqm to use large files.

For more information about file systems, see File system support.

The size of the log file depends on the log settings that you use. The minimum sizes are for circular logging using the default settings. For more information about log sizes, see Calculating the size of the log.

Related concepts:

"Setting up the user and group on AIX" on page 221
On AIX systems, IBM MQ requires a user ID of the name mqm, with a primary group of mqm. The mqm user ID owns the directories and files that contain the resources associated with the product.

Related tasks:

"Configuring and tuning the operating system on AIX"
When installing IBM MQ on AIX systems, there are some additional settings that must be configured.

Configuring and tuning the operating system on AIX

AIX

When installing IBM MQ on AIX systems, there are some additional settings that must be configured.

About this task

When you install IBM MQ on AIX systems, you must configure the following operating system settings:
- File descriptors
- System resource limits

Procedure

- Increase the process limit for the number of file descriptors.

  When running a multi-threaded process such as the agent process, you might reach the soft limit for file descriptors. This limit gives you the IBM MQ reason code MQRC_UNEXPECTED_ERROR (2195) and, if there are enough file descriptors, an IBM MQ FFST file.

  To avoid this problem, increase the process limit for the number of file descriptors. You must alter the nofiles attribute in /etc/security/limits to 10,000 for the mqm user ID, or in the default stanza. To alter the number of file descriptors, complete the following steps:

  1. Check the maximum number of file descriptors available to a process running as mqm:
     ```
     lsuser -a nofiles mqm
     ```
  2. Set the value to at least 10240:
     ```
     chuser nofiles_hard=10240 mqm
     chuser nofiles=10240 mqm
     ```
- Set the system resource limit for data segment and stack segment to unlimited using the following commands in a command prompt:

```
ulimit -d unlimited
ulimit -s unlimited
```

Attention: For an mqm user ID other than root, the value unlimited might not be permitted.

What to do next

You can check your system configuration using the mqconfig command.

For more information on configuring your system, see the technote How to configure UNIX and Linux systems for IBM MQ.

Related concepts:

"Setting up the user and group on AIX" on page 221
On AIX systems, IBM MQ requires a user ID of the name mqm, with a primary group of mqm. The mqm user ID owns the directories and files that contain the resources associated with the product.

"Creating file systems on AIX" on page 223
Before installing IBM MQ, you might need to create file systems for both the product code and working data to be stored. There are minimum storage requirements for these file systems. The default installation directory for the product code can be changed at installation time, but the working data location cannot be changed.

Installing IBM MQ server on AIX

AIX

You can install an IBM MQ server on AIX either interactively or silently.

Before you begin

- Before you start the installation procedure, make sure that you complete the necessary steps that are outlined in "Preparing the system on AIX" on page 221.
- IBM MQ can be installed into System Workload Partitions (WPARs) with both shared and private file systems. For installation into private file systems, IBM MQ can be installed directly into the System WPAR by using the procedure that is outlined in this topic. There are some limitations for shared /usr file systems:
  - The **dspmqinst** and **dspmqver** commands might report the primary installation incorrectly when compared with the symbolic links in /usr/bin. To synchronize the reporting of the primary installation in a System WPAR and the global environment, run **setmqinst** with the **-i** or **-x** parameter, on the individual zones.
  - You cannot change the primary installation within a WPAR. You must change the primary installation through the global environment, which has appropriate write access to /usr/bin.

  Note: During installation to a non-default location, ATTENTION messages that relate to **errupdate** or **trcupdate** are produced. These messages are not errors. However, AIX system trace for IBM MQ is not supported for installations in a non-default location, and IBM MQ trace must be used for problem determination.
- If you install a copy of IBM MQ server for AIX by using Electronic Software Download, obtained from Passport Advantage, you need to:
  - Decompress the tar file, by using the following command:
    ```
    decompress   WS_MQ_V9.0_TRIAL_FOR_AIX_ML.tar.Z
    ```
  - Extract the installation files from the tar file, by using the following command:
    ```
    tar -xvf   WS_MQ_V9.0_TRIAL_FOR_AIX_ML.tar
    ```
  - Use the installation tools **installp** or **smit** to install the IBM MQ server for AIX.

Tip: If you find that the Function keys do not work in SMIT, try pressing Esc and the Function key number to emulate the required Function key.

About this task

IBM MQ is supplied as a set of filesets that are installed by using the standard AIX installation tools. The procedure uses the system management interface tool (SMIT), but you can choose to use **installp**, **geninstall** or the web-based System Manager. You can select which components you want to install. The components and file sets are listed in "IBM MQ components for AIX" on page 216.

This procedure installs IBM MQ into the default location of /usr/mqm.

If you want to install IBM MQ in any one of the following situations:
- As the first installation on your system by using **installp**
- As the first installation on your system, and you are installing the product to a location that is not the default location
- Alongside an existing installation

Use the procedure that is described in "Installing the IBM MQ server silently on AIX" on page 227.

If you want to carry out a side-by-side installation, alongside an existing installation of IBM MQ in the default location, the existing installation must be IBM WebSphere MQ Version 7.0.1, Fix Pack 6, or later.

You must install the second version of the product in a location that is not the default. To create the non-default installation location you must use the **mkusil** command, which is available only from the command line.

You can then use **installp** (see "Installing the IBM MQ server silently on AIX" on page 227), or SMIT if you select the **Relocatable Software Installation** menu item.

If you want to carry out a single stage migration, refer to Single-stage migration from Version 7.0.1 or later to the latest version on UNIX, Linux, and Windows.

Procedure

1. Log in as root, or switch to the superuser by using the **su** command.
2. Set your current directory to the location of the installation file. The location might be the mount point of the DVD, a network location, or a local file system directory.
3. Select the required smit window by using the following sequence:
   ```
   Software Installation and Maintenance
   Install and Update Software
   Install and Update from ALL Available Software
   ```
4. Specify the input directory in the **INPUT device / directory for software** field.
   a. Enter a period character **.**
   b. Press the **Enter** key
5. List the software in the **SOFTWARE to install** field:
   a. Enter **.**
   b. Press **F4**
6. Select the filesets to install from the list. If you require messages in a language different from the language that is specified by the locale that is selected on your system, ensure that you include the appropriate message catalog. Enter **ALL** to install all applicable filesets.
7. View the license agreement:
   a. Change **Preview new LICENSE agreements?** to **yes**
   b. Press **Enter**

8. Accept the license agreements and install IBM MQ:
   a. Change **ACCEPT new license agreements?** to **yes**
   b. Change **Preview new LICENSE agreements?** to **no**
   c. Press **Enter**

What to do next

- If you chose this installation to be the primary installation on the system, you must now set it as the primary installation. Enter the following command at the command prompt:
  ```
  MQ_INSTALLATION_PATH/bin/setmqinst -i -p MQ_INSTALLATION_PATH
  ```
  where *MQ_INSTALLATION_PATH* represents the directory where IBM MQ is installed.

  You can have only one primary installation on a system. If there is already a primary installation on the system, you must unset it before you can set another installation as the primary installation. For more information, see Changing the primary installation.
- You might want to set up the environment to work with this installation. You can use the **setmqenv** or **crtmqenv** command to set various environment variables for a particular installation of IBM MQ. For more information, see setmqenv and crtmqenv.
- If you want to confirm that the installation was successful, you can verify your installation. For more information, see "Verifying an IBM MQ installation on AIX" on page 233.

Related concepts:

"Installation location on Multiplatforms" on page 198
You can install IBM MQ into the default location. Alternatively, you can install into a custom location during the installation process. The location where IBM MQ is installed is known as the *MQ_INSTALLATION_PATH*.

"Multiple installations on UNIX, Linux, and Windows" on page 200
On UNIX, Linux, and Windows, it is possible to have more than one copy of IBM MQ on a system.

"Primary installation on UNIX, Linux, and Windows" on page 202
On systems that support multiple installations of IBM MQ (UNIX, Linux, and Windows), the primary installation is the one to which IBM MQ system-wide locations refer. Having a primary installation is optional, but convenient.

Related tasks:

"Installing the IBM MQ server silently on AIX"
You can carry out a non-interactive installation of the IBM MQ server from the command line using the AIX **installp** command. A non-interactive installation is also known as a silent, or unattended installation.

"Uninstalling or modifying IBM MQ on AIX" on page 247
On AIX, you can uninstall the IBM MQ server or client using the System Management Interface Tool (SMIT) or the **installp** command. You can also modify an installation by uninstalling a subset of the file sets.

Related information:

setmqinst

Changing the primary installation

Installing the IBM MQ server silently on AIX

AIX

You can carry out a non-interactive installation of the IBM MQ server from the command line using the AIX **installp** command. A non-interactive installation is also known as a silent, or unattended installation.

Before you begin

Before you start the installation procedure, make sure that you have completed the necessary steps outlined in "Preparing the system on AIX" on page 221.

Note: During installation, errors relating to **errupdate** or **trcupdate** might occur. This can caused by installing to a non-default location, if so these errors can be safely ignored. However, native trace for IBM MQ is only supported when installed in the default location.

About this task

You can use this method to install to a non-default location, and can select which components you want to install. The components and filesets are listed in "IBM MQ components and features" on page 192.

Procedure

1. Log in as root, or switch to the superuser using the **su** command.
2. Set your current directory to the location of the installation file. The location might be the mount point of the CD, a network location, or a local file system directory.
3. Install the product in one of the following ways:
   - Install the whole product in the default location:
     ```
     installp -acgXYd . all
     ```
   - Install selected file sets in the default location:
     ```
     installp -acgXYd . list of file sets
     ```
   - Install the whole product in a non-default location using the -R flag:
     ```
     installp -R USIL_Directory -acgXYd . all
     ```
   - Install selected file sets in a non-default location using the -R flag:
     ```
     installp -R USIL_Directory -acgXYd . list of file sets
     ```

   where *USIL_Directory* is a directory which exists before the command is run; it must not contain any spaces or usr/mqm. IBM MQ is installed underneath the directory specified. For example, if /USIL1 is specified, the IBM MQ product files are located in /USIL1/usr/mqm. This location is known as the *MQ_INSTALLATION_PATH*.

What to do next

- If you have chosen this installation to be the primary installation on the system, you must now set it as the primary installation. Enter the following command at the command prompt:
  ```
  MQ_INSTALLATION_PATH/bin/setmqinst -i -p MQ_INSTALLATION_PATH
  ```
  where *MQ_INSTALLATION_PATH* represents the directory where IBM MQ is installed.

  You can have only one primary installation on a system. If there is already a primary installation on the system, you must unset it before you can set another installation as the primary installation. For more information, see Changing the primary installation.
- You might want to set up the environment to work with this installation. You can use the **setmqenv** or **crtmqenv** command to set various environment variables for a particular installation of IBM MQ . For more information, see setmqenv and crtmqenv.
- If you want to confirm that the installation was successful, you can verify your installation. See "Verifying an IBM MQ installation on AIX" on page 233, for more information.

Related concepts:

"Multiple installations on UNIX, Linux, and Windows" on page 200
On UNIX, Linux, and Windows, it is possible to have more than one copy of IBM MQ on a system.

"Primary installation on UNIX, Linux, and Windows" on page 202
On systems that support multiple installations of IBM MQ (UNIX, Linux, and Windows), the primary installation is the one to which IBM MQ system-wide locations refer. Having a primary installation is optional, but convenient.

Related tasks:

"Installing IBM MQ server on AIX" on page 225
You can install an IBM MQ server on AIX either interactively or silently.

"Uninstalling or modifying IBM MQ on AIX" on page 247
On AIX, you can uninstall the IBM MQ server or client using the System Management Interface Tool (SMIT) or the **installp** command. You can also modify an installation by uninstalling a subset of the file sets.

Related information:

setmqinst

Changing the primary installation

User Specified Installation Location (USIL)

Converting a trial license on AIX

AIX

Convert a trial license to a full license without reinstalling IBM MQ.

When the trial license expires, the "count-down" displayed by the **strmqm** command informs you the license has expired, and the command does not run.

Before you begin

1. IBM MQ is installed with a trial license.
2. You have access to the installation media of a fully licensed copy of IBM MQ.

About this task

Run the **setmqprd** command to convert a trial license to a full license.

If you do not want to apply a full license to your trial copy of IBM MQ, you can uninstall it at any time.

Procedure

1. Obtain the full license from the fully licensed installation media.

   The full license file is `amqpcert.lic`. On AIX, it is in the `/`*MediaRoot*`/licenses` directory on the installation media.
2. Run the **setmqprd** command from the installation that you are upgrading:

   ```
   MQ_INSTALLATION_PATH/bin/setmqprd /MediaRoot/licenses/amqpcert.lic
   ```

Related information:
setmqprd

Displaying messages in your national language on AIX

AIX

To display messages from a different national language message catalog, you must install the appropriate catalog and set the **LANG** environment variable.

About this task

Messages in the language specified by the locale selected on your machine at installation time are installed by default.

To find out which language is currently in use, run the **locale** command.

If this returns a language which is not one of the national languages provided by IBM MQ, you must select a national language, otherwise you will not get a message catalog installed on your system.

Message catalogs for all languages are installed in *MQ_INSTALLATION_PATH*/msg/*language identifier*, where *language identifier* is one of the identifiers in Table 33. If you require messages in a different language, perform the following steps:

Procedure

1. Install the appropriate message catalog (see "IBM MQ components and features" on page 192).
2. To select messages in a different language, ensure the **LANG** environment variable is set to the identifier for the language you want to install:

Table 33. Language identifiers

Identifier	**Language**
cs_CZ	Czech
de_DE	German
es_ES	Spanish
fr_FR	French
hu_HU	Hungarian
it_IT	Italian
ja_JP	Japanese
ko_KR	Korean
pl_PL	Polish
pt_BR	Brazilian Portuguese
ru_RU	Russian
zh_CN	Simplified Chinese
zh_TW	Traditional Chinese

AIX has some additional message catalogs:

Table 34. AIX specific language identifiers

Identifier	Language
Ja_JP	Japanese
Zh_CN	Simplified Chinese
Zh_TW	Traditional Chinese

Installing an IBM MQ client on AIX

AIX

You can interactively install the IBM MQ client for AIX using `smit`.

Before you begin

Before you start the installation procedure, make sure that you have completed the necessary steps outlined in "Preparing the system on AIX" on page 221.

About this task

IBM MQ is supplied as a set of filesets that are installed using the standard AIX installation tools. The procedure uses the System Management Interface Tool (`smit`), but you can choose to use **`installp`**, **`geninstall`** or the web-based System Manager. You can select which components you want to install. The components and filesets are listed in "IBM MQ components for AIX" on page 216. You must install at least the Runtime and Client components.

This procedure installs IBM MQ into the default location. If you want to install to a non-default location, you must use **`installp`**, see "Installing an IBM MQ client silently on AIX" on page 232.

Procedure

1. Log in as root, or switch to the superuser using the **`su`** command.
2. Make your current directory the location of the installation file. The location might be the mount point of the DVD, a network location, or a local file system directory.
3. Select the required `smit` window using the following sequence:
   ```
   Software Installation and Maintenance
   Install and Update Software
   Install and Update from ALL Available Software
   ```
4. Click **List** to display the input device or directory for the software and select the location that contains the installation images.
5. Select the **SOFTWARE to install** field to obtain a list of available filesets, and select the filesets you want to install. Ensure that you include the appropriate message catalog if you require messages in a language different from the language specified by the locale specified on your system. Enter **ALL** to install all applicable filesets.
6. Change **Preview new LICENSE agreements?** to **yes** and press Enter to view the license agreements.
7. If you have a previous version of the product on your system, change the **Automatically install requisite software** to **no**.
8. Change **ACCEPT new license agreements?** to **yes** and press Enter to accept the license agreements.
9. Change **Preview new LICENSE agreements?** to **no** and press Enter to install IBM MQ.

What to do next

- If you have chosen this installation to be the primary installation on the system, you must now set it as the primary installation. Enter the following command at the command prompt:

```
MQ_INSTALLATION_PATH/bin/setmqinst -i -p MQ_INSTALLATION_PATH
```

You can have only one primary installation on a system. If there is already a primary installation on the system, you must unset it before you can set another installation as the primary installation. For more information, see Changing the primary installation.

- You might want to set up the environment to work with this installation. You can use the **setmqenv** or **crtmqenv** command to set various environment variables for a particular installation of IBM MQ. For more information, see setmqenv and crtmqenv.
- For instructions on how to verify your installation, see "Testing communication between a client and a server on AIX" on page 246.

Related tasks:

"Uninstalling or modifying IBM MQ on AIX" on page 247
On AIX, you can uninstall the IBM MQ server or client using the System Management Interface Tool (SMIT) or the **installp** command. You can also modify an installation by uninstalling a subset of the file sets.

Installing an IBM MQ client silently on AIX

AIX

You can carry out a non-interactive, or silent, installation of an IBM MQ client from the command line using the AIX **installp** command.

Before you begin

Before you start the installation procedure, make sure that you have completed the necessary steps outlined in "Preparing the system on AIX" on page 221.

Note: Installation to a non-default location is not supported on systems that have the AIX Trusted Computing Base (TCB) enabled.

About this task

You can use this method to install to a non-default location, and can select which components you want to install. The components and filesets are listed in "IBM MQ components and features" on page 192. You must install at least the Runtime and Client components.

Procedure

1. Log in as root, or switch to the superuser using the **su** command.
2. Set your current directory to the location of the installation file. The location might be the mount point of the DVD, a network location, or a local file system directory.
3. Install the product in one of the following ways:
   - Install the whole product in the default location:
     ```
     installp -acgXYd . all
     ```
   - Install selected filesets in the default location:
     ```
     installp -acgXYd . list of file sets
     ```
   - Install the whole product in a non-default location using the -R flag:
     ```
     installp -R USIL_Directory -acgXYd . all
     ```
   - Install selected filesets in a non-default location using the -R flag:
     ```
     installp -R USIL_Directory -acgXYd . list of file sets
     ```

   where the directory specified with the `-R` flag is an AIX User Specified Installation Location (USIL) directory which exists before the command is run; it must not contain any spaces or `usr/mqm`.

IBM MQ is installed underneath the directory specified. For example, if `/USIL1` is specified, the IBM MQ product files are located in `/USIL1/usr/mqm`. This location is known as the *MQ_INSTALLATION_PATH*.

What to do next

- If you have chosen this installation to be the primary installation on the system, you must now set it as the primary installation. Enter the following command at the command prompt:

  ```
  MQ_INSTALLATION_PATH/bin/setmqinst -i -p MQ_INSTALLATION_PATH
  ```

  You can have only one primary installation on a system. If there is already a primary installation on the system, you must unset it before you can set another installation as the primary installation. For more information, see Changing the primary installation.
- You might want to set up the environment to work with this installation. You can use the **setmqenv** or **crtmqenv** command to set various environment variables for a particular installation of IBM MQ. For more information, see setmqenv and crtmqenv.
- For instructions on how to verify your installation, see "Testing communication between a client and a server on AIX" on page 246.

Verifying an IBM MQ installation on AIX

AIX

The topics in this section provide instructions on how to verify a server or a client installation of IBM MQ on AIX systems.

About this task

You can verify a local (stand-alone) server installation or a server-to-server installation of the IBM MQ server:
- A local server installation has no communication links with other IBM MQ installations.
- A server-to-server installation does have links to other installations.

You can also verify that your IBM MQ MQI client installation completed successfully and that the communication link is working.

Procedure

- To verify a local server installation, see "Verifying a local server installation on AIX" on page 234.
- To verify a server-to-server installation, see "Verifying a server-to-server installation on AIX" on page 237.
- To verify a client installation, see "Verifying a client installation using the command line on AIX" on page 243.

Verifying a local server installation on AIX

AIX

You can use either the command line or the postcard application to verify a local (stand-alone) installation on HP-UX.

About this task

You can use the command line to verify that IBM MQ is successfully installed, and that the associated communication links are working properly.

You can also verify an installation using the postcard application. The postcard application is Java based and requires a system with the ability to view a graphical display.

Procedure

- To use the command line to verify an installation, see "Verifying a local server installation using the command line on AIX."
- To use the postcard application to verify an installation, see "Verifying a local server installation using the command line on AIX."

Verifying a local server installation using the command line on AIX: AIX

On AIX systems, you can verify a local server installation by using the command line to create a simple configuration of one queue manager and one queue. You can also verify an installation using the postcard application.

Before you begin

To verify the installation, you must first install the samples package.

Before beginning the verification procedure, you might want to check that you have the latest fixes for your system. For more information about where to find the latest updates, see "Checking requirements on AIX" on page 218.

About this task

Use the following steps to configure your default queue manager from the command line. After the queue manager is configured, use the `amqsput` sample program to put a message on the queue. You then use the `amqsget` sample program to get the message back from the queue.

IBM MQ object definitions are case-sensitive. Any text entered as an MQSC command in lowercase is converted automatically to uppercase unless you enclose it in single quotation marks. Make sure that you type the examples exactly as shown.

Procedure

1. On an AIX system, log in as a user in the `mqm` group.
2. Set up your environment:
   a. Set up environment variables for use with a particular installation by entering one the following command:
      ```
      . MQ_INSTALLATION_PATH/bin/setmqenv -s
      ```
      where *`MQ_INSTALLATION_PATH`* refers to the location where IBM MQ is installed.
   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 a queue manager called `QMA` by entering the following command:
   ```
   crtmqm QMA
   ```
   Messages indicate when the queue manager is created, and when the default IBM MQ objects are created.
4. Start the queue manager by entering the following command:
   ```
   strmqm QMA
   ```
   A message indicates when the queue manager starts.
5. Start MQSC by entering the following command:
   ```
   runmqsc QMA
   ```
   A message indicates when MQSC starts. MQSC has no command prompt.
6. Define a local queue called `QUEUE1` by entering the following command:
   ```
   DEFINE QLOCAL (QUEUE1)
   ```
   A message indicates when the queue is created.
7. Stop MQSC by entering the following command:
   ```
   end
   ```
   Messages are shown, followed by the command prompt.

Note: Subsequent steps require that the samples package is installed.

8. Change into the *`MQ_INSTALLATION_PATH`*`/samp/bin` directory, which contains the sample programs. *`MQ_INSTALLATION_PATH`* represents the high-level directory in which IBM MQ is installed.
9. Put a message on the queue by entering the following commands
   ```
   ./amqsput QUEUE1 QMA
   ```
   The following messages are shown:
   ```
   Sample AMQSPUT0 start
   target queue is QUEUE1
   ```
10. Type some message text on one or more lines, where each line is a different message. 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. Get the messages from the queue, by entering the following command:
    ```
    ./amqsget QUEUE1 QMA
    ```
    The sample program starts, and your messages are displayed.

Results

You have successfully verified your local installation.

Verifying a local server installation using the Postcard application on AIX: AIX

Sending messages successfully between two Postcard applications verifies a local installation.

Before you begin

The postcard application is Java based and requires a system with the ability to view a graphical display.

You must ensure that you are a member of the IBM MQ administrators group (`mqm`).

Note: Using Postcard to verify an IBM MQ installation is only possible if there is one IBM MQ installation on that box. The Default Configuration wizard will not create a default configuration if a queue manager already exists on the box. The Default Configuration wizard will run on any installation on a box but only one default configuration can be created per box. Using Postcard to verify second and subsequent installations of IBM MQ on the same box is not possible.

To verify that the local installation is working, you can run two instances of the Postcard application on the same server. The postcard application can send messages to, and receive messages from, other postcard applications. Successful sending and receiving of messages verifies that IBM MQ is installed and working correctly on the server.

Procedure

1. Log on as a user in group `mqm`.
2. Start the postcard application in one of the following ways:
   a. From the command line:
      1) Change the directory to *MQ_INSTALLATION_PATH*/java/bin. *MQ_INSTALLATION_PATH* represents the high-level directory in which IBM MQ is installed.
      2) Run the postcard application by entering the following command:
      ```
      ./postcard
      ```
   b. From the IBM MQ Explorer:
      1) If the Welcome to IBM MQ Explorer Content view page does not show, click **IBM MQ** in the Navigator view to show the Welcome page.
      2) Click **Launch Postcard** to start the Postcard.
3. At the Postcard - Sign On window, type in a nickname to use to send messages within the Postcard application (for example, `User1`).
4. Select the queue manager to use as the mailbox:
   - If you do not have any queue managers, you are prompted to either launch the Default Configuration or close the Postcard application. Launching the Default Configuration creates a default queue manager.
   - If the only queue manager on your server is the default queue manager, this queue manager is used automatically for the postcard application. The default queue manager is created by running the Default Configuration wizard
   - If you have created your own queue managers, but you have not run the Default Configuration wizard, select an appropriate queue manager from the list.
   - If you have run the Default Configuration wizard and you want to use the default queue manager, but there are other queue managers on your server, select the **Advanced** check box. Then select **Use Default Configuration as mailbox**.
   - If you have run the Default Configuration wizard and also created your own queue managers, and you do not want to use the default queue manager, select the **Advanced** check box. Then select **Choose queue manager as mailbox**, and then select the appropriate queue manager from the list.

When your selection is complete, click **OK** to display your first Postcard window.

5. Run a second instance of the Postcard application by following the steps used to open the first instance of the Postcard application.
6. The Postcard - Sign On panel is displayed again. Type in a second nickname to use to send messages within this second Postcard application (for example, `User2`).
7. Repeat the selection of the queue manager that you want to use as the mailbox (as described in step 4). The queue manager you select for this second Postcard must be the same queue manager as used for the first instance of the Postcard application.
8. In the first Postcard, (User1), enter the nickname (`User2`) for the second Postcard application in the **To:** field. Because the sender and receiver are on the same server, you can leave the **On:** field blank.
9. Type a message in the **Message:** field and click **Send**.
10. The **Postcards sent and received** area of the Postcard shows details of the message. In the sending Postcard, the message is displayed as sent. In the receiving Postcard, the message is displayed as received.
11. In the receiving Postcard, (User2), double-click the message in the **Postcards sent and received** area to view it. When this message arrives, it verifies that IBM MQ is correctly installed.

What to do next

Depending on your situation, you might want to do the following tasks:

- Install IBM MQ on other servers. Follow the installation procedure for the appropriate platform. Ensure that you use the **Join Default Cluster** window in the Default Configuration wizard to add the other servers to the cluster on your first server.
- Install the IBM MQ MQI client on other servers.
- Continue with further administration tasks, see Administering IBM MQ.

Verifying a server-to-server installation on AIX

AIX

You can use either the command line or the postcard application to verify a server-to-server installation on AIX.

Before you begin

For a server-to-server verification, the communication links between the two systems must be checked. Before you can do the verification, you must therefore ensure that the communications protocol is installed and configured on both systems.

On AIX, IBM MQ supports both TCP and SNA.

The examples in this task use TCP/IP. If you do not use TCP, see Setting up communication on UNIX and Linux.

About this task

For a server-to server installation, you can use the command line to verify that IBM MQ is successfully installed, and that the associated communication links are working properly.

You can also verify an installation using the postcard application. The postcard application is Java based and requires a system with the ability to view a graphical display.

Procedure

- To use the command line to verify an installation, see “Verifying a server-to-server installation using the command line on AIX.”
- To use the postcard application to verify an installation, see “Verifying a server-to-server installation using the Postcard application on AIX” on page 241.

Verifying a server-to-server installation using the command line on AIX: AIX

You can verify a server-to-server installation using two servers, one as a sender and one as a receiver.

Before you begin

- Make sure that TCP/IP and IBM MQ are installed on both servers (see “Verifying a server-to-server installation on AIX” on page 237).
- Make sure that you are a member of the IBM MQ administrators group (**mqm**) on each server.
- Decide which installation is the sender server and which installation is the receiver server. The installations might be on the same system, or on different systems.

About this task

IBM MQ object definitions are case-sensitive. Any text entered as an MQSC command in lowercase is converted automatically to uppercase unless you enclose it in single quotation marks. Make sure that you type the examples exactly as shown.

Procedure

1. On the **receiver** server:
   a. On AIX, log in as a user in the mqm group.
   b. Check which ports are free, for example by running **netstat**. For more information about this command, see the documentation of your operating system.

      If port 1414 is not in use, make a note of 1414 to use as the port number in step 2 h. Use the same number for the port for your listener later in the verification. If it is in use, note a port that is not in use; for example 1415.
   c. Set up the environment for the installation you are using by entering the following command at the command prompt:
      ```
      . MQ_INSTALLATION_PATH/bin/setmqenv -s
      ```
      where *MQ_INSTALLATION_PATH* refers to the location where IBM MQ is installed.
   d. Create a queue manager called QMB by entering the following command at the command prompt:
      ```
      crtmqm QMB
      ```
      Messages tell you that the queue manager has been created, and that the default IBM MQ objects have been created.
   e. Start the queue manager by entering the following command:
      ```
      strmqm QMB
      ```
      A message tells you when the queue manager has started.
   f. Start MQSC by entering the following command:
      ```
      runmqsc QMB
      ```
      A message tells you that MQSC has started. MQSC has no command prompt.
   g. Define a local queue called RECEIVER.Q by entering the following command:
      ```
      DEFINE QLOCAL (RECEIVER.Q)
      ```

A message tells you the queue has been created.

h. Define a listener by entering the following command:

```
DEFINE LISTENER (LISTENER1) TRPTYPE (TCP) CONTROL (QMGR) PORT ( PORT_NUMBER )
```

Where *port_number* is the name of the port the listener runs on. This number must be the same as the number used when defining your sender channel.

i. Start the listener by entering the following command:

```
START LISTENER (LISTENER1)
```

Note: Do not start the listener in the background from any shell that automatically lowers the priority of background processes.

j. Define a receiver channel by entering the following command:

```
DEFINE CHANNEL (QMA.QMB) CHLTYPE (RCVR) TRPTYPE (TCP)
```

A message tells you when the channel has been created.

k. End MQSC by typing:

```
end
```

Some messages are displayed, followed by the command prompt.

2. On the **sender** server:

a. As the sender server is an AIX system, log in as a user in the `mqm` group.

b. Set up the environment for the installation you are using by entering the following command at the command prompt:

```
. MQ_INSTALLATION_PATH/bin/setmqenv -s
```

where *`MQ_INSTALLATION_PATH`* refers to the location where IBM MQ is installed.

c. Create a queue manager called `QMA` by entering the following command at the command prompt:

```
crtmqm QMA
```

Messages tell you that the queue manager has been created, and that the default IBM MQ objects have been created.

d. Start the queue manager, by entering the following command:

```
strmqm QMA
```

A message tells you when the queue manager has started.

e. Start MQSC by entering the following command:

```
runmqsc QMA
```

A message tells you that an MQSC session has started. MQSC had no command prompt.

f. Define a local queue called `QMB` (to be used as a transmission queue) by entering the following command:

```
DEFINE QLOCAL (QMB) USAGE (XMITQ)
```

A message tells you when the queue has been created.

g. Define a local definition of the remote queue with by entering the following command:

```
DEFINE QREMOTE (LOCAL.DEF.OF.REMOTE.QUEUE) RNAME (RECEIVER.Q) RQMNAME ('QMB') XMITQ (QMB)
```

h. Define a sender channel by entering one of the following commands:

con-name is the TCP/IP address of the receiver system. If both installations are on the same system, the *con-name* is `localhost`. *port* is the port you noted in 1 b. If you do not specify a port, the default value of 1414 is used.

```
DEFINE CHANNEL (QMA.QMB) CHLTYPE (SDR) CONNAME ('CON-NAME(PORT)') XMITQ (QMB) TRPTYPE (TCP)
```

i. Start the sender channel by entering the following command:

```
START CHANNEL(QMA.QMB)
```

The receiver channel on the receiver server starts automatically when the sender channel starts.

j. Stop MQSC by entering the following command:

```
end
```

Some messages are displayed, followed by the command prompt.

k. If the sender server is a UNIX or Linux system, change into the *MQ_INSTALLATION_PATH*/samp/bin directory. This directory contains the sample programs. *MQ_INSTALLATION_PATH* represents the high-level directory in which IBM MQ is installed.

l. If both the sender server and receiver server are installations on the same system, check that the queue managers have been created on different installations by entering the following command:

```
dspmq -o installation
```

If the queue managers are on the same installation, move either QMA to the sender installation or QMB to the receiver installation by using the **setmqm** command. For more information, see setmqm.

m. Put a message on the local definition of the remote queue, which in turn specifies the name of the remote queue. Enter one of the following commands:

- On Windows:

  ```
  amqsput LOCAL.DEF.OF.REMOTE.QUEUE QMA
  ```

- On UNIX and Linux:

  ```
  ./amqsput LOCAL.DEF.OF.REMOTE.QUEUE QMA
  ```

A message tells you that amqsput has started.

n. Type some message text on one or more lines, followed by a blank line. A message tells you that amqsput has ended. Your message is now on the queue and the command prompt is displayed again.

3. On the **receiver** server:

a. As your receiver server is an AIX system, change into the *MQ_INSTALLATION_PATH*/samp/bin directory. This directory contains the sample programs. *MQ_INSTALLATION_PATH* represents the high-level directory in which IBM MQ is installed.

b. Get the message from the queue on the receiver by entering the following command:

```
./amqsget RECEIVER.Q QMB
```

The sample program starts, and your message is displayed. After a pause, the sample ends. Then the command prompt is displayed.

Results

You have now successfully verified the server-to-server installation.

Verifying a server-to-server installation using the Postcard application on AIX: AIX

You can use two instances of the Postcard application to verify that a server-to-server installation is working.

Before you begin

You can use the Postcard application on two servers, one instance of the Postcard application on each server, to verify that a server-to-server installation is working. Successful sending and receiving of messages verifies that IBM MQ is successfully installed, and that communication between the two servers is working correctly.

Note:
- If the system has multiple IBM MQ installations, ensure that Postcard has not been run before on any installations on that server. As the default configuration can only exist on one IBM MQ installation per system, the Default Configuration wizard and Postcard can not be used for verification of a second or any subsequent installation.
- The two server installations must be on different systems to do a server-to-server verification using the postcard application. To verify a server-to-server installation on the same machine, you can use the command line.
- Make sure that TCP/IP and IBM MQ are installed on both servers.
- Make sure that your systems are able to view a graphical display.
- Make sure that you are a member of the IBM MQ administrators group (`mqm`) on each server.
- Check that one of the following scenarios applies:
  - Neither server has had any queue managers created.
  - Use the Default Configuration wizard to create default queue managers on each server and link them to the default cluster.

    Details on how to use the Default Configuration wizard are provided in this topic.
  - Both servers have existing queue managers and these queue managers are in the same cluster.

    If your queue managers are not in the same cluster, create new queue managers on both servers. Then create a cluster, and ensure that the queue managers that you create on each server belong to that cluster.
  - You have configured channels to communicate between the two servers.

    For instructions on how to set up the channels, see "Verifying a server-to-server installation using the command line on AIX" on page 238. After you have set up the channels, follow the instructions in this topic to verify your server-to-server installation.

Procedure
1. On the first server, log on as a user in group `mqm`.
2. Start the postcard application in one of the following ways:
   a. From the command line:
      1) Change the directory to *`MQ_INSTALLATION_PATH`*`/java/bin`. *`MQ_INSTALLATION_PATH`* represents the high-level directory in which IBM MQ is installed.
      2) Run the postcard application by entering the following command:
         ```
         ./postcard
         ```
   b. From the IBM MQ Explorer:
      1) If the Welcome to IBM MQ Explorer Content view page does not show, click **IBM MQ** in the Navigator view to show the Welcome page.
      2) Click **Launch Postcard** to start the Postcard.

3. At the Postcard - Sign On window, type a nickname to use to send messages within the Postcard application. For example, `User1` for the first server, and `User2` for the second server.
4. When you have completed the wizard, you are taken back to the Postcard - Sign On window.
5. Select the queue manager to use as the mailbox:
   - If you do not have any queue managers, you are prompted to either launch the Default Configuration or close the Postcard application. Work through the Default Configuration wizard. When you get to the option to join the queue manager to the default cluster, tick the check box. On the next screen:
     - For the first server, select **yes, make it the repository for the cluster**.
     - For the second server, select **No another computer has already joined the cluster as a repository**. When requested, enter the location of the repository, by typing the name of the sender server.
   - If the only queue manager on your server is the default queue manager, this queue manager is used automatically for the postcard application. The default queue manager is created by running the Default Configuration wizard
   - If you have created your own queue managers, but you have not run the Default Configuration wizard, select an appropriate queue manager from the list.
   - If you have run the Default Configuration wizard and you want to use the default queue manager, but there are other queue managers on your server, select the **Advanced** check box. Then select **Use Default Configuration as mailbox**.
   - If you have run the Default Configuration wizard and also created your own queue managers, and you do not want to use the default queue manager, select the **Advanced** check box. Then select **Choose queue manager as mailbox**, and then select the appropriate queue manager from the list.

   When your selection is complete, click **OK**.
6. Complete steps 1 - 5 for the second server.
7. In the Postcard on the first server:
   a. Enter the nickname (`user2`) for the Postcard application on the second server in the **To:** field.
   b. Enter the queue manager on the second server in the **On:** field.
   c. Type a message in the **Message:** field and click **Send**.
8. In the Postcard on the second server:
   a. In the **Postcards sent and received**, double-click the message marked as received to view the message from the first server.
   b. Optional: Send a postcard to the first server by adapting the instructions in step 7. You must enter details of the first server in the **To:** field and the **On:** field.

   The messages verify that IBM MQ is correctly installed and that your communication link between the two servers is working correctly.

Verifying a client installation using the command line on AIX

AIX

You can verify a client installation using the command line. On the server you create a queue manager, a local queue, a listener, and a server-connection channel. You must also apply security rules to allow the client to connect and make use of the queue defined. On the client you create a client-connection channel, and then use the sample PUT and GET programs to complete the verification procedure.

About this task

The verification procedure shows how to create a queue manager called `queue.manager.1`, a local queue called `QUEUE1`, and a server-connection channel called `CHANNEL1` on the server.

It shows how to create the client-connection channel on the IBM MQ MQI client workstation. It then shows how to use the sample programs to put a message onto a queue, and get the message from the queue.

The example does not address any client security issues. See Setting up IBM MQ MQI client security for details if you are concerned with IBM MQ MQI client security issues.

The verification procedure assumes that:
- The full IBM MQ server product has been installed on a server.
- The server installation is accessible on your network.
- The IBM MQ MQI client software has been installed on a client system.
- The IBM MQ sample programs have been installed.
- TCP/IP has been configured on the server and client systems. For more information, see Configuring connections between the server and client.

Procedure

1. Set up the server using the command line, using the instructions in "Setting up the server using the command line on AIX."
2. Set up the client, using the instructions in "Connecting to a queue manager, using the `MQSERVER` environment variable on AIX" on page 245.
3. Test the communications between client and server, using the instructions in "Testing communication between a client and a server on AIX" on page 246.

Setting up the server using the command line on AIX: AIX

Follow these instructions to create a queue manager, queue, and channel on the server. You can then use these objects to verify the installation.

About this task

These instructions assume that no queue manager or other IBM MQ objects have been defined.

IBM MQ object definitions are case-sensitive. Any text entered as an MQSC command in lowercase is converted automatically to uppercase unless you enclose it in single quotation marks. Make sure that you type the examples exactly as shown.

Procedure

1. Create a user ID on the server that is not in the `mqm` group. This user ID must exist on the server and client. This is the user ID that the sample applications must be run as, otherwise a 2035 error is returned.

2. Log in as a user in the mqm group.
3. You must set various environment variables so that the installation can be used in the current shell. You can set the environment variables by entering the following command:

```
. MQ_INSTALLATION_PATH/bin/setmqenv -s
```

   where *MQ_INSTALLATION_PATH* refers to the location where IBM MQ is installed.
4. Create a queue manager called QUEUE.MANAGER.1 by entering the following command:
   ```
   crtmqm QUEUE.MANAGER.1
   ```
   You see messages telling you that the queue manager has been created.
5. Start the queue manager by entering the following command:
   ```
   strmqm QUEUE.MANAGER.1
   ```
   A message tells you when the queue manager has started.
6. Start MQSC by entering the following command:
   ```
   runmqsc QUEUE.MANAGER.1
   ```
   A message tells you that an MQSC session has started. MQSC has no command prompt.
7. Define a local queue called QUEUE1 by entering the following command:
   ```
   DEFINE QLOCAL(QUEUE1)
   ```
   A message tells you when the queue has been created.
8. Allow the user ID that you created in step 1 to use QUEUE1 by entering the following command:
   ```
   SET AUTHREC PROFILE(QUEUE1) OBJTYPE(QUEUE) PRINCIPAL(' non_mqm_user ') AUTHADD(PUT,GET)
   ```
   where *non_mqm_user* is the user ID created in step 1. A message tells you when the authorization has been set. You must also run the following command to give the user ID authority to connect:
   ```
   SET AUTHREC OBJTYPE(QMGR) PRINCIPAL(' non_mqm_user ') AUTHADD(CONNECT)
   ```
   If this command is not run, a 2305 stop error is returned.
9. Define a server-connection channel by entering the following command:
   ```
   DEFINE CHANNEL (CHANNEL1) CHLTYPE (SVRCONN) TRPTYPE (TCP)
   ```
   A message tells you when the channel has been created.
10. Allow your client channel to connect to the queue manager and run under the user ID that you created in step 1, by entering the following MQSC command:
    ```
    SET CHLAUTH(CHANNEL1) TYPE(ADDRESSMAP) ADDRESS(' client_ipaddr ') MCAUSER(' non_mqm_user ')
    ```
    where *client_ipaddr* is the IP address of the client system, and *non_mqm_user* is the user ID created in step 1. A message tells you when the rule has been set.
11. Define a listener by entering the following command:
    ```
    DEFINE LISTENER (LISTENER1) TRPTYPE (TCP) CONTROL (QMGR) PORT (port_number)
    ```
    where *port_number* is the number of the port the listener is to run on. This number must be the same as the number used when defining your client-connection channel in "Installing an IBM MQ client on AIX" on page 231.

    Note: If you omit the port parameter from the command, a default value of 1414 is used for the listener port. If you want to specify a port other than 1414, you must include the port parameter in the command, as shown.
12. Start the listener by entering the following command:
    ```
    START LISTENER (LISTENER1)
    ```

13. Stop MQSC by entering:
    ```
    end
    ```

    You see some messages, followed by the command prompt.

What to do next

Follow the instructions to set up the client. See "Connecting to a queue manager, using the MQSERVER environment variable on AIX."

Connecting to a queue manager, using the MQSERVER environment variable on AIX: AIX

When an IBM MQ application is run on the IBM MQ MQI client, it requires the name of the MQI channel, the communication type, and the address of the server to be used. Provide these parameters by defining the MQSERVER environment variable.

Before you begin

Before you start this task, you must complete the task, "Setting up the server using the command line on AIX" on page 243, and save the following information:
- The host name or IP address of the server and port number that you specified when creating the listener.
- The channel name of the server-connection channel.

About this task

This task describes how to connect an IBM MQ MQI client, by defining the MQSERVER environment variable on the client.

You can give the client access to the generated client channel definition table, amqclchl.tab instead; see Accessing client-connection channel definitions.

Procedure
1. Log in as the userid that you created in Step 1 of "Verifying a client installation using the command line on AIX" on page 243.
2. Check the TCP/IP connection. From the client, enter one of the following commands:
   - `ping server-hostname`
   - `ping n.n.n.n`

     `n.n.n.n` represents the network address. You can set the network address in IPv4 dotted decimal form, for example, `192.0.2.0`. Alternatively, set the address in IPv6 hexadecimal form, for example `2001:0DB8:0204:acff:fe97:2c34:fde0:3485`.

   If the **ping** command fails, correct your TCP/IP configuration.
3. Set the MQSERVER environment variable. From the client, enter the following command:
   ```
   export MQSERVER=CHANNEL1/TCP/'server-address (port)'
   ```

   Where:
   - *CHANNEL1* is the server-connection channel name.
   - *server-address* is the TCP/IP host name of the server.
   - *port* is the TCP/IP port number the server is listening on.

   If you do not give a port number, IBM MQ uses the one specified in the qm.ini file, or the client configuration file. If no value is specified in these files, IBM MQ uses the port number identified in the TCP/IP services file for the service name MQSeries. If an MQSeries entry in the services file does

not exist, a default value of 1414 is used. It is important that the port number used by the client and the port number used by the server listener program are the same.

What to do next

Use the sample programs to test communication between the client and server; see "Testing communication between a client and a server on AIX."

Testing communication between a client and a server on AIX: AIX

On the IBM MQ MQI client workstation, use the amqsputc sample program to put a message on the queue at the server workstation. Use the amqsgetc sample program to get the message from the queue back to the client.

Before you begin

Complete the previous topics in this section:
- Set up a queue manager, channels, and queue.
- Open a command window.
- Set system environment variables.

About this task

Note that IBM MQ object definitions are case-sensitive. Text entered as an MQSC command in lowercase is converted automatically to uppercase unless you enclose it in single quotation marks. Make sure that you type the examples exactly as shown.

Procedure

1. Change to the *MQ_INSTALLATION_PATH*/samp/bin directory, which contains the sample programs. *MQ_INSTALLATION_PATH* represents the high-level directory in which IBM MQ is installed.
2. You must set certain environment variables so that the installation can be used in the current shell. You can set the environment variables by entering the following command:
   ```
   . MQ_INSTALLATION_PATH/bin/setmqenv -s
   ```
   where *MQ_INSTALLATION_PATH* refers to the location where IBM MQ is installed.
3. Start the PUT program for QUEUE1 on QUEUE.MANAGER.1 by entering the following command:
   ```
   ./amqsputc QUEUE1 QUEUE.MANAGER.1
   ```
   If the command is successful, the following messages are displayed:
   ```
   Sample AMQSPUT0 start target queue is QUEUE1
   ```
   Tip: You might get the error, MQRC_NOT_AUTHORIZED (2035). By default, channel authentication is enabled when a queue manager is created. Channel authentication prevents privileged users accessing a queue manager as an IBM MQ MQI client. For verifying the installation, you can either change the MCA user ID to a non-privileged user, or disable channel authentication. To disable channel authentication run the following MQSC command:
   ```
   ALTER QMGR CHLAUTH(DISABLED)
   ```
   When you finish the test, if you do not delete the queue manager, re-enable channel authentication:
   ```
   ALTER QMGR CHLAUTH(ENABLED)
   ```
4. Type some message text, then press **Enter** twice. The following message is displayed:
   ```
   Sample AMQSPUT0 end
   ```
   Your message is now on the queue that is on the server queue manager.

5. Start the GET program for `QUEUE1` on `QUEUE.MANAGER.1` by entering the following command:

   ```
   ./amqsgetc QUEUE1 QUEUE.MANAGER.1
   ```

   The sample program starts, and your message is displayed. After a short pause (approximately 30 seconds), the sample ends and the command prompt is displayed again.

Results

You have now successfully verified the client installation.

What to do next

1. You must set various environment variables on the server so that the installation can be used in the current shell. You can set the environment variables by entering the following command:

   ```
   . MQ_INSTALLATION_PATH/bin/setmqenv -s
   ```

   where *MQ_INSTALLATION_PATH* refers to the location where IBM MQ is installed.
2. On the server, stop the queue manager by entering the following command:

   ```
   endmqm QUEUE.MANAGER.1
   ```
3. On the server, delete the queue manager by entering the following command:

   ```
   dltmqm QUEUE.MANAGER.1
   ```

Uninstalling or modifying IBM MQ on AIX

AIX

On AIX, you can uninstall the IBM MQ server or client using the System Management Interface Tool (SMIT) or the **installp** command. You can also modify an installation by uninstalling a subset of the file sets.

Before you begin

If any updates have been applied, remove them before starting the uninstallation or modification procedure. For more information, see Reverting to the previous maintenance level on AIX.

Important: You must stop all IBM MQ queue managers, other objects, and applications, before you begin the process to uninstall or modify IBM MQ.

Procedure

1. Stop all IBM MQ applications associated with the installation you are uninstalling or modifying, if you have not already done so.
2. For a server installation, end any IBM MQ activity associated with the installation you are uninstalling or modifying:
   a. Log in as a user in the group `mqm`.
   b. Set up your environment to work with the installation you want to uninstall or modify. Enter the following command:

      ```
      . MQ_INSTALLATION_PATH/bin/setmqenv
      ```

      where *MQ_INSTALLATION_PATH* refers to the location where IBM MQ is installed.
   c. Display the state of all queue managers on the system. Enter the following command:

      ```
      dspmq -o installation
      ```
   d. Stop all running queue managers associated with the installation you want to uninstall or modify. Enter the following command for each queue manager:

      ```
      endmqm QMgrName
      ```

   e. Stop any listeners associated with the queue managers. Enter the following command for each queue manager:

      ```
      endmqlsr -m QMgrName
      ```
3. Log in as root.
4. Uninstall or modify IBM MQ using either **installp** or **smit**. If IBM MQ was installed in a non-default location, you must use **installp**.
   - To uninstall or modify IBM MQ by using **installp**, enter one of the following commands:
     - To uninstall an installation in the default location `/usr/mqm`:

       ```
       installp -u mqm
       ```
     - To uninstall an installation in a non-default location:

       ```
       installp -R usil -u mqm
       ```

       where *usil* is the path of the User Specified Installation Location (USIL) specified when the product was installed.
     - To modify an installation in a non-default location:

       ```
       installp -R usil -u list of file sets
       ```

       where *usil* is the path of the User Specified Installation Location (USIL) specified when the product was installed.
   - To uninstall or modify IBM MQ by using **smit**, complete the following steps:
     a. Select the required **smit** window using the following sequence:

        ```
        Software Installation and Maintenance
        Software Maintenance and Utilities
        Remove Installed Software
        ```
     b. List the software in the **SOFTWARE name** field:
        1) Enter **.**
        2) Press **F4**
     c. Select the file sets to uninstall from the list (those beginning with mqm):
        - For a complete uninstall, select all file sets.
        - To modify the installation, select a subset of the file sets.

        After selecting the file sets, press **Enter**. There is an option at this stage to do a preview. Leave the option set to the default value of **Yes** to preview the file sets you are uninstalling, or select **No** to not preview these file sets.
     d. Press **Enter** on the **Remove Installed Software** panel, it asks whether you are sure, press **Enter**.

Results

After uninstallation, certain files under the directory trees `/var/mqm` and `/etc/opt/mqm` are not removed. These files contain user data and remain so subsequent installations can reuse the data. Most of the remaining files contain text, such as INI files, error logs, and FDC files. The directory tree `/var/mqm/shared` contains files that are shared across installations, including the executable shared libraries `libmqzsd.a` and `libmqzsd_r.a`.

What to do next

- If the product successfully uninstalled, you can delete any files and directories contained in the `/usr/mqm` directory under the User Specified Installation Location (USIL) specified in the **installp** uninstallation command.
- Use the **lslpp** command to check for other products installed in the USIL. If there are no other products installed in the USIL and you do not intend to use it again, you can delete the USIL using the **rmusil** command.

- If there are no other IBM MQ installations on the system, and you are not planning to reinstall or migrate, you can delete the `/var/mqm` and `/etc/opt/mqm` directory trees, including the files `libmqzsd.a` and `libmqzsd_r.a`. Deleting these directories destroys all queue managers and their associated data.

Installing and uninstalling IBM MQ client for HP Integrity NonStop Server

NSS Client

Tasks that are associated with installing the IBM MQ client on HP Integrity NonStop Server systems are grouped in this section.

About this task

To prepare for installation and to install the IBM MQ client, complete the following tasks.

For information about how to uninstall the IBM MQ client, see “Uninstalling the IBM MQ client for HP Integrity NonStop Server” on page 263.

If product fixes or updates are made available, see IBM MQ maintenance tasks for information about how to apply these changes.

Procedure

1. Check the system requirements. See “Hardware and software requirements on HP Integrity NonStop Server systems” on page 250.
2. Plan your installation. See “Planning your client installation on HP Integrity NonStop Server” on page 252.
3. Prepare your system for installation of the IBM MQ client for HP Integrity NonStop Server. See “Setting up the user and group on HP Integrity NonStop Server” on page 256.
4. Install an IBM MQ client. See “Installing the IBM MQ client on HP Integrity NonStop Server” on page 257.
5. Verify your installation. See “Verifying an IBM MQ installation on HP Integrity NonStop Server” on page 258.

IBM MQ client components for HP Integrity NonStop Server

NSS Client

There are no optional components within the client installer when you install the IBM MQ client for HP Integrity NonStop Server.

An installation of the IBM MQ client for HP Integrity NonStop Server contains product binary files, command utilities, and samples. There are no optional components.

Related concepts:

"Planning your client installation on HP Integrity NonStop Server" on page 252
This section describes what to do to prepare your HP Integrity NonStop Server system for installing the IBM MQ client for HP Integrity NonStop Server.

"Hardware and software requirements on HP Integrity NonStop Server systems"
Check that the server environment meets the prerequisites for installing the IBM MQ client for HP Integrity NonStop Server. Check the product readme files and install missing prerequisite software supplied on the server CD.

Related information:

Planning your IBM MQ client environment on HP Integrity NonStop Server

Hardware and software requirements on HP Integrity NonStop Server systems

NSS Client

Check that the server environment meets the prerequisites for installing the IBM MQ client for HP Integrity NonStop Server. Check the product readme files and install missing prerequisite software supplied on the server CD.

Hardware

The IBM MQ client for HP Integrity NonStop Server typically requires certain hardware specifications to run:

- HP Integrity NonStop Server H and J series
- Two or more processors
- At least 1 GB, and ideally 4 GB of memory per processor
- 500 MB of free disk space in the Guardian and OSS file systems

Operating system

Two operating systems are supported by the IBM MQ client for HP Integrity NonStop Server:

- HP Integrity NonStop Server running H06.24 or later NonStop OS
- HP Integrity NonStop BladeSystem running J06.13 or later NonStop OS

You must be running one of these operating systems to install the IBM MQ client for HP Integrity NonStop Server.

Other software requirements

IBM MQ client for HP Integrity NonStop Server has some additional software requirements:

- The operating system software, Open System Services (OSS), must be active, with file systems and a local sockets subsystem that is configured and running.
- Safeguard must be active.
- If two-phase commit transaction support is required, then TMF must be active and Pathway must be configured and available. The connected queue manager must be at IBM WebSphere MQ Version 7.1 or later.
- If the Java Message Service (JMS) API is required, then HP Integrity NonStop Server for Java V6 must be available.
- You might require compatible compilers, linkers, and maybe other tools for the C, C++, COBOL, JMS, or pTAL languages if you want to build and use applications.

File system requirements

In the selected installation root directory, in the OSS file system, an installation creates:

- `opt` - a directory tree that contains the "static" files for an installation in OSS.
- `var` - a directory tree that contains the "variable" files for an installation in OSS.

An installation also creates a single subvolume in the Guardian file system, which is selected during installation.

Related concepts:

"License requirements" on page 194
You must have purchased sufficient licenses for your installation. The details of the license agreement is stored on your system at installation time so that you can read it at any time. IBM MQ supports IBM License Metric Tool (ILMT).

Related tasks:

"Checking requirements on Windows" on page 444
Before you install IBM MQ on Windows, you must check for the latest information and system requirements.

Related information:

UNIX Linux Disk space requirements

Verifying system software prerequisites on HP Integrity NonStop Server

NSS Client

Use the HP Integrity NonStop Server TACL utility, SYSINFO, to verify the base OS level of the HP Integrity NonStop Server.

Procedure

From a TACL command prompt, enter **SYSINFO**.

Results

The system information is displayed as shown in the following example:

```
SYSINFO - T9268H01 - (01 OCT 2004)  SYSTEM \NODE1  Date 05 Nov 2010, 11:56:51
Copyright 2003 Hewlett-Packard Development Company, L.P.

         System name    \NODE1
  EXPAND node number    025
       Current SYSnn    SYS00
       System number    nnnnnn
 Software release ID    J06.10.00
```

In this example, the base OS level is J06.10.00.

What to do next

Compare the base OS level with the "Hardware and software requirements on HP Integrity NonStop Server systems" on page 250. Verify any other HP Integrity NonStop Server software prerequisites or recommendations identified in the documentation or the product README; for example, SPRs to particular products.

Planning your client installation on HP Integrity NonStop Server

▶ NSS Client

This section describes what to do to prepare your HP Integrity NonStop Server system for installing the IBM MQ client for HP Integrity NonStop Server.

Understanding multiple installations

The IBM MQ client for HP Integrity NonStop Server can be installed more than once on an HP Integrity NonStop Server system. In addition, multiple different versions of the client can be installed on a single HP Integrity NonStop Server system, and be maintained independently.

To install IBM MQ client for HP Integrity NonStop Server, you must specify two locations; one in the OSS file system, and one in the Guardian file system, which is used by the installer to store the results of the installation. These locations must not contain or overlap with any other IBM MQ client for HP Integrity NonStop Server installation. The locations must also be free of other files.

Each installation is independent and self-contained, with all data, such as configuration logs, or trace and program files located within the installation directory hierarchy. All commands and libraries use an embedded runtime search path (RPath) to ensure that they load their dependencies from the same installation.

As several installations might be present, each application must locate and load the IBM MQ client libraries from the correct installation.

- For native applications, an application that is linked with the IBM MQ MQIC.LIB installation library inherits the IBM MQ installation RPATH, and can run without environment variables. Environment variables in OSS, for example, *_RLD_LIB_PATH* or DEFINEs in Guardian, are only required if you want to run the application using a different IBM MQ installation.
- For Java applications using the Java Messaging Service (JMS) API, the client Java archive (JAR) must be from the correct installation, and must be included on the class path. For more information, see Environment variables used by IBM MQ classes for JMS.

Product packaging and delivery

IBM MQ client for HP Integrity NonStop Server is downloaded to the OSS file system as a single file.

The IBM MQ client for HP Integrity NonStop Server package file is a self-extracting archive (SFX) that contains an installer and all files that are required to create installations.

The SFX for IBM MQ client for HP Integrity NonStop Server has a file extension of `.run`. There is no concept of placed files. When run, the SFX creates a single installation, directly from the archive, into the OSS and Guardian file systems.

The SFX can be used to create as many installations of the IBM MQ client for HP Integrity NonStop Server as you require. No information about installations is retained in the SFX, and no tools are provided for extracting individual files from the SFX.

File system on HP Integrity NonStop Server

NSS Client

Before you install the IBM MQ client for HP Integrity NonStop Server, make sure that the file system is set up correctly.

Review "Hardware and software requirements on HP Integrity NonStop Server systems" on page 250 to make sure that you understand the approximate amount of disk space in the OSS and Guardian file systems that is required for an installation. The OSS file set that is used for the installation requires enough free space for the installation files and the files you create in the installation. The Guardian volume that you use for installation does not require auditing.

Work with your systems administrator to verify the OSS file set and Guardian file system storage requirements, at least for an initial estimate of the storage. The best way to determine more precisely how much storage you would eventually need in production is to produce a prototype configuration and model the message storage requirements, scaling up as necessary for your production system.

OSS file system objects

For the OSS file system objects, this section concentrates on the differences between the HP Integrity NonStop Server installation, and the standard UNIX installation. Multiple independent installations are supported.

The `opt` and `var` trees must be present in a common root directory, which is selected at installation time. The opt tree contains files that do not change. For example, this tree contains program, library, dll, header files, and "static" data files. The `var` tree contains files that might change, and do hold status about the installation itself. Examples of files that this tree holds are configuration files, and log files.

Both the `opt` and `var` directories contain a single directory named `mqm`. The content of both trees is rooted in the `opt/mqm` and `var/mqm` directories.

Table 35. Summary of the contents at the top-level of `opt/mqm`

Directory	Purpose	Contents
`bin`	Contains the OSS programs and libraries for an installation	• `G` is a symbolic link file that locates the Guardian installation subvolume • `amq*` files, containing the product executables for the client • `lib*` files, containing product dll files • Files containing control commands, and other utilities and scripts
`inc`	Contains the header files for building IBM MQ applications	• `.h` files, which are C language header files • `.tal` files which are pTAL header files • `.cpy` files which are COBOL copy file • `cobcpy32` and `cobcpy64` directories for the individual COBOL copy files

Table 35. Summary of the contents at the top-level of `opt/mqm` (continued)

Directory	**Purpose**	**Contents**
`lib`	Contains the import libraries needed to link applications	• `G` is a symbolic link file that locates the Guardian installation subvolume • `amq*` files, containing product dll files • `iconv` is a directory that contains data conversion tables • `lib*` files, which are product dll files • `mqicb` used for supplying to a CONSULT directive for compiling COBOL programs
`license`	Contains text versions of the IBM License for the IBM MQ client for HP Integrity NonStop Server, which is translated into each supported national language	• `Lic_.txt` files, which are the individual national language translations of the license. • `notices.txt` is a file that contains any additional license terms from non-IBM software that is included with IBM MQ , if any.
`mq.id`	Single file that contains information about the build level and installation package	All contents in this directory might be used by IBM Support personnel.
`msg`	Contains globalization files for use by IBM MQ , in logging and displaying output in the supported national language translations	The contents include: • `amq.cat` globalization message catalog currently in use by the installation, created by the OSS utility "gencat" • `amq.msg` unprocessed globalization data that is used as input by `gencat` to create the catalog • Other minor files and directories that support the different translations

Table 35. Summary of the contents at the top-level of `opt/mqm` (continued)

Directory	Purpose	Contents
`samp`	Contains sample code and executables to illustrate the use of IBM MQ	• `*.cbl` sample COBOL language source files • `*.c` sample 'C' language source files • `*.tal` sample pTAL language source files • `ccsid.new` backup file of `ccsid.tbl` • `ccsid.tbl` file that contains a table of supported CCSIDs • `*.ini` sample configuration files • `java` directory that contains source for sample Java applications • `jms` directory that contains source for sample JMS applications • `bin` directory that contains executable versions of the samples • `dlq` directory that contains a source for the sample Dead Letter Queue Handler • `preconnect` directory that contains source for preconnect exit

For more information about the samples that are provided, see Samples for the IBM MQ client for HP Integrity NonStop Server.

Table 36. Summary of the contents at the top-level of `var/mqm`

Directory	Purpose	Contents
`conv`	Contains data conversion files	Binary data that supports the data conversion function for IBM MQ
`errors`	Contains installation-wide error logs and FDC files	Standard content, for example: • `AMQERR01.LOG` - current system-wide error log file • `AMQERR02.LOG` - previous system-wide error log file • `AMQERR03.LOG` - oldest system-wide error log file • `*.FDC` FFST files
`exits`	Stores DLLs containing exit code that is loaded by the queue managers in the installation	This file is empty at installation
`log`	Contains log files for recording and controlling units of work	Standard content
`mqs.ini`	The installation configuration file	Standard content
`qmgrs`	Directory beneath the location where all of the queue manager directories are created	Standard content
`sockets`	Directory tree that contains various queue manager control files	Standard content

Table 36. Summary of the contents at the top-level of `var/mqm` (continued)

Directory	Purpose	Contents
`trace`	Defined location to which trace data is written by IBM MQ	Standard content

Guardian installation subvolume

The Guardian single installation subvolume contains both the programs and libraries needed at runtime.

Table 37. Contents of the Guardian installation subvolume

File	Description
`AMQINST`	Internal file that describes installation configuration
`AMQS*`	Samples that are built for Guardian
`B*SAMP`	Sample build files for the various supported languages
`CMQ*`	Header files for the various supported languages, where files ending in: • `h` are C headers • `T` are pTAL headers • `L` are COBOL headers
`MQ*`	Product libraries
`MQS*C`	Sample C language source files
`MQS*T`	Sample pTAL language source files
`MQS*L`	Sample COBOL language source files

Control commands are also included, for a list, see IBM MQ client for HP Integrity NonStop Server client commands.

Setting up the user and group on HP Integrity NonStop Server

NSS Client

The administrator user ID must be used to administer the IBM MQ client for HP Integrity NonStop Server.

Ensure that you have access to an IBM MQ client for HP Integrity NonStop Server user ID in the user group called MQM. The MQM group must be created before the client can be installed. All user IDs that are used to install the client must have MQM as their primary group. If this user group does not exist, or you do not have access to such a user, contact your systems administrator.

Installing the IBM MQ client on HP Integrity NonStop Server

NSS Client

Installing the IBM MQ client on a HP Integrity NonStop Server system.

Before you begin

Before you start the installation procedure, make sure that you complete the necessary steps that are outlined in "Setting up the user and group on HP Integrity NonStop Server" on page 256.

About this task

After preparing your system for installation, install the IBM MQ client for HP Integrity NonStop Server by following the instructions. After installation, you might want to verify your installation to check that it installed successfully. There are three steps to the installation:

1. Downloading the installation package.
2. Running the installer.
3. Setting the environment.

Procedure

1. Log in to the OSS user ID that owns the installation. The OSS user ID must have MQM as its primary group.
2. Download the installation package file. Ensure that you use "binary mode" when you download the installation package file to OSS. Any corruption in the file causes the self-extracting archive to fail to run. After you have downloaded the package file, ensure that it has read and execute permissions for the user ID that is installing the package.
3. Set the *_RLD_FIRST_LIB_PATH* variable to `MQ_INSTALLATION_PATH/opt/mqm/bin`
4. Optional: Make your current directory the location of the installation file.
5. Type the following command to start the interactive installation procedure:

   `./name of package file -i OSS install_root -g Guardian install_root`

   where

   name of package file> is the name of the installation package.

   OSS install_root is the OSS root directory of the new installation.

   Guardian install_root is the Guardian subvolume for the new installation.

   Both `-i` and `-g` options are mandatory.

   - `-i` specifies the new or empty OSS directory that contains the `opt/mqm` and `var/mqm` directories of the installation.
   - `-g` specifies the subvolume into which the Guardian components of the IBM MQ client for HP Integrity NonStop Server are installed. The Guardian subvolume can be specified in either OSS-form or Guardian-form and can be abbreviated. The Guardian subvolume specification is not case sensitive. The following examples show valid Guardian subvolume specifications:
     - `/G/vol/subvol`
     - `vol/subvol`
     - `\$VOL.SUBVOL`
     - `vol.subvol`
6. Optional: For OSS, set your environment by installing the binary files into your path. To do this, type the following command:

   `export PATH=$PATH:OSS_install_root/opt/mqm/bin`

   where *OSS_install_root* is the OSS root directory of the new installation.

Example

To install the IBM MQ client for HP Integrity NonStop Server from package `mat1.run`, type the following command:

```
./mat1.run -i ~install/mq75client -g /G/data04/mqm
```

The command installs the OSS components into new `opt/mqm` and `var/mqm` directories in `~install/mq75client`. It installs the Guardian components into `/G/data04/mqm`.

What to do next

For instructions on how to verify your installation, see "Testing communication between a client and a server on HP Integrity NonStop Server" on page 261.

Verifying an IBM MQ installation on HP Integrity NonStop Server

NSS Client

The topics in this section provide instructions on how to verify a client installation of IBM MQ on HP Integrity NonStop Server systems.

Verifying a client installation using the command line on HP Integrity NonStop Server

NSS Client

You can verify a client installation using the command line. On the server you create a queue manager, a local queue, a listener, and a server-connection channel. You must also apply security rules to allow the client to connect and make use of the queue defined. On the client you create a client-connection channel, and then use the sample PUT and GET programs to complete the verification procedure.

The verification procedure shows how to create a queue manager called `queue.manager.1`, a local queue called `QUEUE1`, and a server-connection channel called `CHANNEL1` on the server.

It shows how to create the client-connection channel on the IBM MQ MQI client workstation. It then shows how to use the sample programs to put a message onto a queue, and get the message from the queue.

The example does not address any client security issues. See Setting up IBM MQ MQI client security for details if you are concerned with IBM MQ MQI client security issues.

The verification procedure assumes that:
- The server installation is accessible on your network.
- The IBM MQ MQI client software has been installed on a client system.
- The IBM MQ sample programs have been installed.
- TCP/IP has been configured on the server and client systems. For more information, see Configuring connections between the server and client.

First, set up the server using the command line, using the instructions in "Setting up the server using the command line on HP Integrity NonStop Server" on page 259.

Once you have set up the server, you must set up the client, using the instructions in "Connecting to a queue manager, using the `MQSERVER` environment variable on HP Integrity NonStop Server" on page 260.

Finally, you can test the communications between client and server, using the instructions in "Testing communication between a client and a server on HP Integrity NonStop Server" on page 261.

Setting up the server using the command line on HP Integrity NonStop Server: NSS Client

Follow these instructions to create a queue manager, queue, and channel on the server. You can then use these objects to verify the installation.

About this task

These instructions assume that no queue manager or other IBM MQ objects have been defined.

IBM MQ object definitions are case-sensitive. Any text entered as an MQSC command in lowercase is converted automatically to uppercase unless you enclose it in single quotation marks. Make sure that you type the examples exactly as shown.

Procedure

1. Create a user ID on the server that is not in the `mqm` group. This user ID must exist on the server and client. This is the user ID that the sample applications must be run as, otherwise a 2035 error is returned.
2. Log in as a user in the mqm group.
3. You must set various environment variables so that the installation can be used in the current shell. You can set the environment variables by entering the following command:

```
. MQ_INSTALLATION_PATH/bin/setmqenv -s
```

   where *`MQ_INSTALLATION_PATH`* refers to the location where IBM MQ is installed.
4. Create a queue manager called `QUEUE.MANAGER.1` by entering the following command:
   ```
   crtmqm QUEUE.MANAGER.1
   ```
   You see messages telling you that the queue manager has been created.
5. Start the queue manager by entering the following command:
   ```
   strmqm QUEUE.MANAGER.1
   ```
   A message tells you when the queue manager has started.
6. Start MQSC by entering the following command:
   ```
   runmqsc QUEUE.MANAGER.1
   ```
   A message tells you that an MQSC session has started. MQSC has no command prompt.
7. Define a local queue called `QUEUE1` by entering the following command:
   ```
   DEFINE QLOCAL(QUEUE1)
   ```
   A message tells you when the queue has been created.
8. Allow the user ID that you created in step 1 to use `QUEUE1` by entering the following command:
   ```
   SET AUTHREC PROFILE(QUEUE1) OBJTYPE(QUEUE) PRINCIPAL(' non_mqm_user ') AUTHADD(PUT,GET)
   ```
   where *non_mqm_user* is the user ID created in step 1. A message tells you when the authorization has been set. You must also run the following command to give the user ID authority to connect:
   ```
   SET AUTHREC OBJTYPE(QMGR) PRINCIPAL(' non_mqm_user ') AUTHADD(CONNECT)
   ```
   If this command is not run, a 2305 stop error is returned.
9. Define a server-connection channel by entering the following command:
   ```
   DEFINE CHANNEL (CHANNEL1) CHLTYPE (SVRCONN) TRPTYPE (TCP)
   ```
   A message tells you when the channel has been created.

10. Allow your client channel to connect to the queue manager and run under the user ID that you created in step 1, by entering the following MQSC command:
    ```
    SET CHLAUTH(CHANNEL1) TYPE(ADDRESSMAP) ADDRESS(' client_ipaddr ') MCAUSER(' non_mqm_user ')
    ```
    where *client_ipaddr* is the IP address of the client system, and *non_mqm_user* is the user ID created in step 1. A message tells you when the rule has been set.
11. Define a listener by entering the following command:
    ```
    DEFINE LISTENER (LISTENER1) TRPTYPE (TCP) CONTROL (QMGR) PORT (port_number)
    ```
    where *port_number* is the number of the port the listener is to run on. This number must be the same as the number used when defining your client-connection channel in “Installing the IBM MQ client on HP Integrity NonStop Server” on page 257.

    Note: If you omit the port parameter from the command, a default value of `1414` is used for the listener port. If you want to specify a port other than `1414`, you must include the port parameter in the command, as shown.
12. Start the listener by entering the following command:
    ```
    START LISTENER (LISTENER1)
    ```
13. Stop MQSC by entering:
    ```
    end
    ```
    You see some messages, followed by the command prompt.

What to do next

Follow the instructions to set up the client. See “Connecting to a queue manager, using the `MQSERVER` environment variable on HP Integrity NonStop Server.”

Connecting to a queue manager, using the `MQSERVER` environment variable on HP Integrity NonStop Server: NSS Client

When an IBM MQ application is run on the IBM MQ MQI client, it requires the name of the MQI channel, the communication type, and the address of the server to be used. Provide these parameters by defining the `MQSERVER` environment variable.

Before you begin

Before you start this task, you must complete the task, “Setting up the server using the command line on HP Integrity NonStop Server” on page 259, and save the following information:
- The host name or IP address of the server and port number that you specified when creating the listener.
- The channel name of the server-connection channel.

About this task

This task describes how to connect an IBM MQ MQI client, by defining the `MQSERVER` environment variable on the client.

You can give the client access to the generated client channel definition table, `amqclchl.tab` instead; see Accessing client-connection channel definitions.

Procedure

1. Log in as the userid that you created in Step 1 of “Setting up the server using the command line on HP Integrity NonStop Server” on page 259.
2. Check the TCP/IP connection. From the client, enter one of the following commands:
   - `ping server-hostname`
   - `ping n.n.n.n`

     `n.n.n.n` represents the network address. You can set the network address in IPv4 dotted decimal form, for example, `192.0.2.0`. Alternatively, set the address in IPv6 hexadecimal form, for example `2001:0DB8:0204:acff:fe97:2c34:fde0:3485`.

   If the **`ping`** command fails, correct your TCP/IP configuration.
3. Set the `MQSERVER` environment variable. From the client, enter one of the following commands:
   a. On IBM MQ client for HP Integrity NonStop Server:
   ```
   export MQSERVER=CHANNEL1/TCP/' server-address (port)'
   ```
   b. On HP Integrity NonStop Server Guardian systems:
   ```
   param MQSERVER CHANNEL1/TCP/server-address (port)
   ```
   Where:
   - *CHANNEL1* is the server-connection channel name.
   - *server-address* is the TCP/IP host name of the server.
   - *port* is the TCP/IP port number the server is listening on.

   If you do not give a port number, IBM MQ uses the one specified in the `qm.ini` file, or the client configuration file. If no value is specified in these files, IBM MQ uses the port number identified in the TCP/IP services file for the service name `MQSeries`. If an `MQSeries` entry in the services file does not exist, a default value of `1414` is used. It is important that the port number used by the client and the port number used by the server listener program are the same.

What to do next

Use the sample programs to test communication between the client and server; see “Testing communication between a client and a server on HP Integrity NonStop Server.”

Testing communication between a client and a server on HP Integrity NonStop Server: NSS Client

On the IBM MQ MQI client workstation, use the amqsputc sample program to put a message on the queue at the server workstation. Use the amqsgetc sample program to get the message from the queue back to the client.

Before you begin

Complete the previous topics in this section:
- Set up a queue manager, channels, and queue.
- Open a command window.
- Set system environment variables.

About this task

Note that IBM MQ object definitions are case-sensitive. Text entered as an MQSC command in lowercase is converted automatically to uppercase unless you enclose it in single quotation marks. Make sure that you type the examples exactly as shown.

Procedure

1. Change into the *MQ_INSTALLATION_PATH*/opt/mqm/samp/bin directory, which contains the sample programs. . *MQ_INSTALLATION_PATH* represents the high-level directory in which IBM MQ is installed.
2. Start the PUT program for QUEUE1 on QUEUE.MANAGER.1 by entering the following command:

   ```
   ./amqsputc QUEUE1 QUEUE.MANAGER.1
   ```

   If the command is successful, the following messages are displayed:

   ```
   Sample AMQSPUT0 start target queue is QUEUE1
   ```

   Tip: You might get the error, MQRC_NOT_AUTHORIZED (2035). By default, channel authentication is enabled when a queue manager is created. Channel authentication prevents privileged users accessing a queue manager as an IBM MQ MQI client. For verifying the installation, you can either change the MCA user ID to a non-privileged user, or disable channel authentication. To disable channel authentication run the following MQSC command:

   ```
   ALTER QMGR CHLAUTH(DISABLED)
   ```

   When you finish the test, if you do not delete the queue manager, re-enable channel authentication:

   ```
   ALTER QMGR CHLAUTH(ENABLED)
   ```
3. Type some message text, then press **Enter** twice. The following message is displayed:

   ```
   Sample AMQSPUT0 end
   ```

   Your message is now on the queue that is on the server queue manager.
4. Start the GET program for QUEUE1 on QUEUE.MANAGER.1 by entering the following command:

   ```
   ./amqsgetc QUEUE1 QUEUE.MANAGER.1
   ```

   The sample program starts, and your message is displayed. After a short pause (approximately 30 seconds), the sample ends and the command prompt is displayed again.

Results

You have now successfully verified the client installation.

What to do next

1. You must set various environment variables on the server so that the installation can be used in the current shell. You can set the environment variables by entering the following command:

   ```
   . MQ_INSTALLATION_PATH/bin/setmqenv -s
   ```

   where *MQ_INSTALLATION_PATH* refers to the location where IBM MQ is installed.
2. On the server, stop the queue manager by entering the following command:

   ```
   endmqm QUEUE.MANAGER.1
   ```
3. On the server, delete the queue manager by entering the following command:

   ```
   dltmqm QUEUE.MANAGER.1
   ```

Uninstalling the IBM MQ client for HP Integrity NonStop Server

NSS Client

On HP Integrity NonStop Server systems, you can uninstall the IBM MQ client by using the **rm** command.

Procedure

1. Stop all IBM MQ applications that are associated with the installation you are uninstalling.
2. Log in to the OSS as the user ID that owns the installation.
3. Use the OSS **rm** command to delete the files from the Guardian subvolume used by the installation. For example, use the following command:
   ```
   rm -rf MQ_INSTALLATION_PATH/opt/mqm/bin/G/*
   ```
4. Use the OSS **rm** command to delete the OSS directory trees for the installation. For example, use the following command:
   ```
   rm -rf MQ_INSTALLATION_PATH
   ```

Installing and uninstalling IBM MQ on HP-UX

HP-UX

Installation tasks that are associated with installing IBM MQ on HP-UX systems are grouped in this section.

About this task

To prepare for installation and to install the IBM MQ components, complete the following tasks.

For information about how to uninstall IBM MQ, see "Uninstalling or modifying IBM MQ on HP-UX" on page 296.

If product fixes or updates are made available, see IBM MQ maintenance tasks for information about how to apply these changes.

Procedure

1. Check the system requirements. See "Checking requirements on HP-UX" on page 266.
2. Plan your installation.
   - As part of the planning process, you must choose which components to install and where to install them. See "IBM MQ components for HP-UX" on page 264.
   - You must also make some platform-specific choices. See "Planning to install IBM MQ on HP-UX" on page 267.
3. Prepare your system for installation of IBM MQ. See "Preparing the system on HP-UX" on page 268.
4. Install IBM MQ server. See "Installing IBM MQ server on HP-UX" on page 274.
5. Optional: Install an IBM MQ client. See "Installing an IBM MQ client on HP-UX" on page 280.
6. Verify your installation. See "Verifying an IBM MQ installation on HP-UX" on page 282.

IBM MQ components for HP-UX

HP-UX

You can select the components that you require when you install IBM MQ.

Important: See IBM MQ license information for details of what each purchase of IBM MQ entitles you to install.

Table 38 shows the components that are available when installing an IBM MQ server or client on an HP-UX system:

Table 38. IBM MQ components for HP-UX systems

Component	**Description**	**Server media**	**Client media**	**Component name**
Runtime	Contains files that are common to both server and client installations. **Note:** This component must be installed.	✓	✓	MQSERIES.MQM-RUNTIME
Server	You can use the server to run queue managers on your system and connect to other systems over a network. Provides messaging and queuing services to applications, and support for IBM MQ client connections.	✓		MQSERIES.MQM-SERVER
Standard Client	The IBM MQ MQI client is a small subset of IBM MQ, without a queue manager, that uses the queue manager and queues on other (server) systems. It can be used only when the system it is on is connected to another system that is running a full server version of IBM MQ. The client and the server can be on the same system if required.	✓	✓	MQSERIES.MQM-CL-HPUX
SDK	The SDK is required for compiling applications. It includes sample source files, and the bindings (files .H, .LIB, .DLL, and others), that you need to develop applications to run on IBM MQ.	✓	✓	MQSERIES.MQM-BASE
Sample programs	The sample application programs are needed if you want to check your IBM MQ installation using the verification procedures.	✓	✓	MQSERIES.MQM-SAMPLES
Java messaging	The files needed for messaging using Java (includes Java Message Service).	✓	✓	MQSERIES.MQM-JAVA
Man pages	UNIX man pages, in U.S. English, for: control commands MQI calls MQSC commands	✓	✓	MQSERIES.MQM-MAN
Java JRE	A Java Runtime Environment that is used by those parts of IBM MQ that are written in Java.	✓	✓	MQSERIES.MQM-JAVAJRE
Message Catalogs	For available languages, see the table of message catalogs that follows.	✓	✓	
IBM Global Security Kit	IBM Global Security Kit V8 Certificate and TLS, Base Runtime.	✓	✓	MQSERIES.MQM-GSKIT

Table 38. IBM MQ components for HP-UX systems (continued)

Component	Description	Server media	Client media	Component name
Managed File Transfer	MQ Managed File Transfer transfers files between systems in a managed and auditable way, regardless of file size or the operating systems used. For information about the function of each component, see Managed File Transfer product options.	✓		MQSERIES.MQM-FTAGENT MQSERIES.MQM-FTBASE MQSERIES.MQM-FTLOGGER MQSERIES.MQM-FTSERVICE MQSERIES.MQM-FTTOOLS
Advanced Message Security	Provides a high level of protection for sensitive data flowing through the IBM MQ network, while not impacting the end applications. You must install this component on all IBM MQ installations that host queues you want to protect. You must install the IBM Global Security Kit component on any IBM MQ installation that is used by a program that puts or gets messages to or from a protected queue, unless you are using only Java client connections. You must install the **Java JRE** component to install this component.	✓		MQSERIES.MQM-AMS
V 9.0.0 AMQP Service	Install this component to make AMQP channels available. AMQP channels support MQ Light APIs. You can use AMQP channels to give AMQP applications access to the enterprise-level messaging facilities provided by IBM MQ.	✓		MQSERIES.MQM-AMQP

Table 39. IBM MQ message catalogs for HP-UX systems

Message catalog language	Component name
Brazilian Portuguese	MQSERIES.MQM-MC-PORT
Czech	MQSERIES.MQM-MC-CZECH
French	MQSERIES.MQM-MC-FRENCH
German	MQSERIES.MQM-MC-GERMAN
Hungarian	MQSERIES.MQM-MC-HUNGARIAN
Italian	MQSERIES.MQM-MC-ITALIAN
Japanese	MQSERIES.MQM-MC-JAPAN
Korean	MQSERIES.MQM-MC-KOREAN
Polish	MQSERIES.MQM-MC-POLISH
Russian	MQSERIES.MQM-MC-RUSSIAN
Spanish	MQSERIES.MQM-MC-SPANISH
Simplified Chinese	MQSERIES.MQM-MC-CHINES
Traditional Chinese	MQSERIES.MQM-MC-CHINET
U.S. English	not applicable

Related concepts:

"IBM MQ components and features" on page 192
You can select the components or features that you require when you install IBM MQ.

"Planning considerations for installation on Multiplatforms" on page 196
Before you install IBM MQ, you must choose which components to install and where to install them. You must also make some platform-specific choices.

Checking requirements on HP-UX

▶ HP-UX

Before you install IBM MQ on HP-UX, you must check for the latest information and system requirements.

About this task

A summary of the tasks that you must complete to check system requirements are listed here with links to further information.

Procedure

1. Check that you have the latest information, including information on hardware and software requirements. See "Where to find product requirements and support information" on page 195.
2. Check that your systems meet the initial hardware and software requirements for HP-UX. See "Hardware and software requirements on HP-UX systems."

   The supported hardware and software environments are occasionally updated. See System Requirements for IBM MQ for the latest information.
3. Check that your systems have sufficient disk space for the installation. See Disk space requirements.
4. Check that you have the correct licenses. See "License requirements" on page 194 and IBM MQ license information.

What to do next

When you have completed these tasks, you are ready to start preparing your system for installation. For the next steps in installing IBM MQ, see "Preparing the system on HP-UX" on page 268.

Related concepts:

"IBM MQ installation overview" on page 192
An overview of concepts and considerations for installing IBM MQ, with links to instructions on how to install, verify, and uninstall IBM MQ on each of the supported platforms.

Related information:

IBM MQ maintenance tasks

Hardware and software requirements on HP-UX systems

▶ HP-UX

Before you install IBM MQ , check that your system meets the hardware and operating system software requirements for the particular components you intend to install.

For hardware and software requirements, see System Requirements for IBM MQ.

IBM MQ does not support host names that contain spaces. If you install IBM MQ on a system with a host name that contains spaces, you are unable to create any queue managers.

Java Message Service and SOAP transport

If you want to use Java Message Service and SOAP support, you need an IBM Java 7 SDK and Runtime Environment Version 7.0 or later.

V 9.0.0 Java 8 is bundled with IBM MQ Version 9.0 but client components are built with Java 7 compatibility flags on.

For development, a JDK is required, and a JRE is required for running. The JRE does not need to be the JRE installed with IBM MQ, but has to be one from the supported list.

For a list of supported JDKs, see System Requirements for IBM MQ.

For further information about SOAP with IBM MQ , see IBM MQ transport for SOAP.

On HP-UX : To run a 64-bit or 32-bit JVM use the -d64 or -d32 parameters on the command line when running a Java application to ensure the correct JVM is used.

You can check the version installed using the following command:

```
java -version
```

Transport Layer Security (TLS)

If you want to use the TLS support, you need the IBM Global Security Kit (GSKit) V8 package. This package is supplied with IBM MQ as one of the components available for installation.

HP-UX
To use TLS, IBM MQ clients on HP-UX must be built using POSIX threads.

Related concepts:

IBM i "Hardware and software requirements on IBM i systems" on page 299
Check that the server environment meets the prerequisites for installing IBM MQ for IBM i. Check the product readme files and install missing prerequisite software supplied on the server CD.

"Hardware and software requirements on Windows systems" on page 445
Check that the server environment meets the prerequisites for installing IBM MQ for Windows and install any prerequisite software that is missing from your system from the server DVD.

Related tasks:

"Checking requirements on Windows" on page 444
Before you install IBM MQ on Windows, you must check for the latest information and system requirements.

Planning to install IBM MQ on HP-UX

HP-UX

Before you install IBM MQ on HP-UX, you must choose which components to install and where to install them. You must also make some platform-specific choices.

About this task

The following steps provide links to additional information to help you with planning your installation of IBM MQ on HP-UX.

As part of your planning activities, make sure that you review the information on hardware and software requirements for the platform on which you are planning to install IBM MQ. For more information, see “Checking requirements on HP-UX” on page 266.

Procedure

1. Decide which IBM MQ components and features to install. See “IBM MQ components and features” on page 192.

   Important: Ensure that your enterprise has the correct license, or licenses, for the components that you are going to install. For more information, see “License requirements” on page 194 and IBM MQ license information.
2. Review the options for naming your installation. In some cases, you can choose an installation name to use instead of the default name. See “Installation name on UNIX, Linux, and Windows” on page 197.
3. Review the options and restrictions for choosing an installation location for IBM MQ. For more information, see “Installation location on Multiplatforms” on page 198.
4. If you plan to install multiple copies of IBM MQ, see “Multiple installations on UNIX, Linux, and Windows” on page 200.
5. If you already have a primary installation, or plan to have one, see “Primary installation on UNIX, Linux, and Windows” on page 202.
6. Make sure that the communications protocol needed for server-to-server verification is installed and configured on both systems that you plan to use. For more information, see “Server-to-server links on UNIX, Linux, and Windows” on page 210.

Preparing the system on HP-UX

HP-UX

On HP-UX systems, you might have to complete several tasks before you install IBM MQ. You might also want to complete other tasks, depending on your installation intentions.

About this task

The tasks that you perform to prepare your systems for installation are listed here. Complete the appropriate tasks for your platform before installing.

Procedure

1. Set up a user ID of the name `mqm`, with a primary group of `mqm`. See “Setting up the user and group on HP-UX” on page 269.
2. Create file systems for both the product code and working data to be stored. See “Creating file systems on HP-UX” on page 270.
3. Configure any additional settings needed for your HP-UX system. See “Configuring and tuning the operating system on HP-UX” on page 271.

What to do next

When you have completed the tasks to prepare the system, you are ready to start installing IBM MQ. To install a server, see “Installing IBM MQ server on HP-UX” on page 274. To install a client, see “Installing an IBM MQ client on HP-UX” on page 280.

Related information:
Planning
Maintaining and migrating
IBM MQ maintenance tasks

Setting up the user and group on HP-UX

HP-UX

On HP-UX systems, IBM MQ requires a user ID of the name `mqm`, with a primary group of `mqm`. The `mqm` user ID owns the directories and files that contain the resources associated with the product.

Creating the user ID and groups

Set the primary group of the `mqm` user to the group `mqm`.

If you are installing IBM MQ on multiple systems you might want to ensure each UID and GID of `mqm` has the same value on all systems. If you are planning to configure multi-instance queue managers, it is essential the UID and GID are the same from system to system. It is also important to have the same UID and GID values in virtualization scenarios.

HP-UX
The user ID value for user `mqm` must be less than 60,000 to avoid problems with the maintenance update process.

You can use the System Management Homepage (SMH), or the **`groupadd`** and **`useradd`** commands to work with user IDs.

Adding existing user IDs to the group

If you want to run administration commands, for example **`crtmqm`** (create queue manager) or **`strmqm`** (start queue manager), your user ID must be a member of the `mqm` group. This user ID must not be longer than 12 characters.

Users do not need `mqm` group authority to run applications that use the queue manager; it is needed only for the administration commands.

Log files created by MQ Telemetry service

The **`umask`** setting of the user ID that creates a queue manager will determine the permissions of the Telemetry log files generated for that queue manager. Even though the ownership of the log files will be set to `mqm`.

Related concepts:

"Creating file systems on AIX" on page 223
Before installing IBM MQ, you might need to create file systems for both the product code and working data to be stored. There are minimum storage requirements for these file systems. The default installation directory for the product code can be changed at installation time, but the working data location cannot be changed.

"Configuring and tuning the operating system on HP-UX" on page 271
Before you install IBM MQ on an HP-UX system, you must check that the kernel is configured correctly.

"Configuring and tuning the operating system on Linux" on page 337
Use this topic when you are configuring IBM MQ on Linux systems.

Related tasks:

"Configuring and tuning the operating system on AIX" on page 224
When installing IBM MQ on AIX systems, there are some additional settings that must be configured.

Related information:

"Configuring and tuning the operating system on Solaris" on page 409
Configure Solaris systems with the resource limits required by IBM MQ.

Creating file systems on HP-UX

HP-UX

Before installing IBM MQ, you might need to create file systems for both the product code and working data to be stored. There are minimum storage requirements for these file systems. The default installation directory for the product code can be changed at installation time, but the working data location cannot be changed.

Determining the size of a server installations file system

To determine the size of the `/var/mqm` file system for a server installation, consider:

- The maximum number of messages in the system at one time.
- Contingency for message buildups, if there is a system problem.
- The average size of the message data, plus 500 bytes for the message header.
- The number of queues.
- The size of log files and error messages.
- The amount of trace that is written to the `/var/mqm/trace` directory.

Storage requirements for IBM MQ also depend on which components you install, and how much working space you need. For more details, see Disk space requirements.

Creating a file system for the working data

Before you install IBM MQ, create and mount a file system called `/var/mqm` which is owned by the user `mqm` in the group `mqm` ; see "Setting up the user and group on Linux" on page 334. This file system is used by all installations of IBM MQ on a system. If possible, use a partition strategy with a separate volume for the IBM MQ data. This means that other system activity is not affected if a large amount of IBM MQ work builds up. Configure the directory permissions to permit the `mqm` user to have full control, for example, file mode 755. These permissions will then be updated during the IBM MQ installation to match the permissions required by the queue manager.

Creating separate file systems for errors and logs

You can also create separate file systems for your log data (`/var/mqm/log`) and error files (`/var/mqm/errors`). If possible, place these directories on different physical disks from the queue manager data (`/var/mqm/qmgrs`) and from each other.

If you create separate file systems the `/var/mqm/errors` directory can be NFS mounted. However, if you choose to NFS-mount `/var/mqm/errors`, the error logs might be lost if the network fails.

You can protect the stability of your queue manager by having separate file systems for:
- `/var/mqm/errors`
- `/var/mqm/trace`
- `/var/mqm/qmgrs`
- `/var/mqm/log`

In the case of `/var/mqm/errors`, it is rare that this directory receives large quantities of data. But it is sometimes seen, particularly if there is a severe system problem leading to IBM MQ writing a lot of diagnostic information in to `.FDC` files. In the case of `/var/mqm/trace`, files are only written here when you use **`strmqtrc`** to start tracing IBM MQ.

You can obtain better performance of normal IBM MQ operations (for example, syncpoints, MQPUT, MQGET of persistent messages) by placing the following on separate disks:
- `/var/mqm/qmgrs`
- `/var/mqm/log`

In the rare event that you need to trace an IBM MQ system for problem determination, you can reduce performance impact by placing the `/var/mqm/trace` file system on a separate disk.

If you are creating separate file systems, allow a minimum of 30 MB of storage for `/var/mqm`, 100 MB of storage for `/var/mqm/log`, and 10 MB of storage for `/var/mqm/errors`. The 100 MB minimum allowance of storage for `/var/mqm/log` is the absolute minimum required for a single queue manager and is not a recommended value. The size of a file system must be scaled according to the number of queue managers that you intend to use, the number of pages per log file, and the number of log files per queue manager.

If you want to use individual queues that hold more than 2 GB of data, you must enable `/var/mqm` to use large files.

For more information about file systems, see File system support.

The size of the log file depends on the log settings that you use. The minimum sizes are for circular logging using the default settings. For more information about log sizes, see Calculating the size of the log.

Related concepts:

"Setting up the user and group on HP-UX" on page 269
On HP-UX systems, IBM MQ requires a user ID of the name `mqm`, with a primary group of `mqm`. The `mqm` user ID owns the directories and files that contain the resources associated with the product.

"Configuring and tuning the operating system on HP-UX"
Before you install IBM MQ on an HP-UX system, you must check that the kernel is configured correctly.

Configuring and tuning the operating system on HP-UX

HP-UX

Before you install IBM MQ on an HP-UX system, you must check that the kernel is configured correctly.

Kernel configuration

It is possible that the default kernel configuration is not adequate because IBM MQ uses semaphores and shared memory.

Before installation, review the configuration of the machine and increase the values if necessary. Consider using the values of the tunable kernel parameters given in Table 40. These values might need to be increased if you obtain any First Failure Support Technology™ (FFST) records.

Note:
1. Semaphore and swap usage do not vary significantly with message rate or message persistence.
2. IBM MQ queue managers are independent of each other. Therefore system tunable kernel parameters, for example `shmmni`, `semmni`, `semmns`, and `semmnu` need to allow for the number of queue managers in the system.

See the HP-UX documentation for information about changing these values.

Table 40. Minimum tunable kernel parameters values

Name	**Value**	**Increase**	**Description**
`shmmax`	268435456	No	Maximum size of a shared-memory segment (bytes)
`shmseg`	1024	No	Maximum number of shared memory segments per process
`shmmni`	1024	Yes	Maximum number of shared memory segments
`semaem`	128	No	Maximum undo value for a semaphore for a single process
`semvmx`	32767	No	Maximum value of a semaphore
`semmns`	4096	Yes	Maximum number of semaphores
`semmni`	128	Yes	Maximum number of semaphore sets
`semmnu`	16384	Yes	Maximum number of process having semaphore operations that can be undone
`semume`	32	No	Maximum number of semaphore undo operations per process
`max_thread_proc`	66	No	Maximum number of threads in a process
`maxfiles`	10000	No	Maximum number of file handles per process (soft limit)
`maxfiles_lim`	10000	No	Maximum number of file handles per process (hard limit)

Notes:
- These values are sufficient to run two moderate sized queue managers on the system. If you intend to run more than two queue managers, or the queue managers are to process a significant workload, you might need to increase the values displayed as *Yes* in the *Increase* column.
- You must restart the system after you change any of the tunable kernel parameters.

System resource limits

You can set global limits for the size of process data segments and the size of process stack segments for the whole system. These limits are set by altering the tunable kernel parameters.

The tunable kernel parameters are:

Parameter	What it controls	Consider minimum value
maxdsiz	Maximum size of the data segment for 32-bit processes	1073741824
maxdsiz_64bit	Maximum size of the data segment for 64-bit processes	1073741824
maxssiz	Maximum size of the stack segment for 32-bit processes	8388608
maxssiz_64bit	Maximum size of the stack segment for 64-bit processes	8388608

If other software on the same machine needs higher values, then the operation of IBM MQ is not adversely affected if those higher values are used.

For the full documentation for these parameters see the HP-UX product documentation.

To apply the settings to an HP-UX 11i system which has the System Administration Manager (SAM) utility, you can use SAM to achieve the following steps:
- Select and alter the parameters
- Process the new kernel
- Apply the changes and restart the system

Other releases of HP-UX might provide different facilities to set the tunable kernel parameters. Consult your HP-UX product documentation for the relevant information.

The `ulimit` shell command

On a per-shell basis, the available limits can be tuned down from the values stored for the "System resource limits" on page 272 preceding parameters. Use the **`ulimit`** shell command to tune the values of the parameters with a combination of the following switches:

Switch	Meaning
-H	The hard limit
-S	The soft limit
-d	The data segments size
-s	The stack segment size

Verifying that the kernel settings are applied

You can verify that the resource limits have not been lowered by a **`ulimit`** command and that the queue manager has the correct limits. To verify the limits, go to the shell from which the queue manager is started and enter the following command:

```
ulimit -Ha
ulimit -Sa
```

Among the console output you see:

```
data(kbytes)  1048576
stack(kbytes) 8192
```

If the lowered numbers are returned, then a **`ulimit`** command has been issued in the current shell to reduce the limits. Consult with your system administrator to resolve the issue.

You can check your system configuration using the mqconfig command.

For more information on configuring your system, see How to configure UNIX and Linux systems for IBM MQ.

Related concepts:

"Setting up the user and group on HP-UX" on page 269
On HP-UX systems, IBM MQ requires a user ID of the name `mqm`, with a primary group of `mqm`. The `mqm` user ID owns the directories and files that contain the resources associated with the product.

"Creating file systems on AIX" on page 223
Before installing IBM MQ, you might need to create file systems for both the product code and working data to be stored. There are minimum storage requirements for these file systems. The default installation directory for the product code can be changed at installation time, but the working data location cannot be changed.

Installing IBM MQ server on HP-UX

HP-UX

You can install an IBM MQ server on HP-UX either interactively or silently.

Before you begin

- Before you start the installation procedure, make sure that you have completed the necessary steps outlined in "Preparing the system on HP-UX" on page 268.
- If you install a copy of IBM MQ server for HP-UX using Electronic Software Download, obtained from Passport Advantage, you need to decompress the `tar.gz` file, and extract the installation files from the tar file, by using the following command:

  ```
  tar -xvf   WS_MQ_V8.0_TRIAL_FOR_HP-UX_ML.tar
  ```

  Important: You must use GNU tar (also known as `gtar`) to unpack any `tar` images.

About this task

This task describes the installation of a server, using the `swinstall` program to select which components you want to install. The components are listed in "IBM MQ components for HP-UX" on page 264.

Note: If you are using a screen reader, use the non-interactive installation option "Installing the IBM MQ server silently on HP-UX" on page 276, so that you can accept the license without viewing it.

If you are installing IBM MQ from a depot that contains service update packages, read Applying maintenance level updates on IBM MQ on HP-UX before installing the service update packages.

Procedure

1. Log in as root, or switch to the superuser using the **su** command.
2. Set your current directory to the location of the installation file. The location might be the mount point of the DVD, a network location, or a local file system directory.
3. Accept the license by running the `mqlicense` script:

   ```
   ./mqlicense.sh
   ```

   The license is displayed. If you accept the license, you can continue the installation.
4. Start the interactive installation procedure by typing the following command,

   ```
   swinstall -s /installation_file
   ```

   /installation_file is the absolute path to the installation file. The path must begin with a / and end with the name of the installation file. The installation file has a file name extension of `.v11`. In the resulting menu screen, select **MQSERIES**.
   a. If you do not want to install all IBM MQ components, open **MQSERIES**

   1) Mark the components you want to install. The installer resolves dependencies automatically.
   2) Review the information displayed by the installer.
5. Optional: To install IBM MQ to a non-default location, select **MQSERIES** from the lower part of the user interface, then select **Actions > Change Product Location**. The default installation location is `/opt/mqm`.

   For each installation, all of the IBM MQ components that you require must be installed in the same location.

   The installation path specified must either be an empty directory, the root of an unused file system, or a path that does not exist. The length of the path is limited to 256 bytes and must not contain spaces.

   Note: Ensure that you do not select **Actions > Change Target** by accident, they are not the same.
6. If this installation is not the first installation on the system, select **Options > Allow creation of multiple versions**
7. Select **Actions > Install**. The log file tells you if there are any problems that need fixing.
8. Fix any problems, then click **OK** to install.

   The system tells you when the installation has finished.
9. If this installation is not the first installation on the system, you must enter the following command to configure IBM MQ.

   Note: *MQ_INSTALLATION_PATH* is the path where you have just installed IBM MQ and the character defining the path is a lower case L.

   ```
   swconfig -x allow_multiple_versions=true MQSERIES,l=MQ_INSTALLATION_PATH
   ```

   If you do not enter this command, the **swlist** command reports the installation as installed instead of configured. You must not use IBM MQ unless the installation is configured.

What to do next

- If you have chosen this installation to be the primary installation on the system, you must now set it as the primary installation. Enter the following command at the command prompt:

  ```
  MQ_INSTALLATION_PATH/bin/setmqinst -i -p MQ_INSTALLATION_PATH
  ```

  where *MQ_INSTALLATION_PATH* represents the directory where IBM MQ is installed.

  You can have only one primary installation on a system. If there is already a primary installation on the system, you must unset it before you can set another installation as the primary installation. For more information, see Changing the primary installation.
- You might want to set up the environment to work with this installation. You can use the **setmqenv** or **crtmqenv** command to set various environment variables for a particular installation of IBM MQ . For more information, see setmqenv and crtmqenv.
- If you want to confirm that the installation was successful, you can verify your installation. See "Verifying an IBM MQ installation on HP-UX" on page 282, for more information.

Related concepts:

"Multiple installations on UNIX, Linux, and Windows" on page 200
On UNIX, Linux, and Windows, it is possible to have more than one copy of IBM MQ on a system.

"Primary installation on UNIX, Linux, and Windows" on page 202
On systems that support multiple installations of IBM MQ (UNIX, Linux, and Windows), the primary installation is the one to which IBM MQ system-wide locations refer. Having a primary installation is optional, but convenient.

Related tasks:

"Installing the IBM MQ server silently on HP-UX"
You can perform a non-interactive installation of the IBM MQ server using the **swinstall** command. A non-interactive installation is also known as a silent, or unattended installation.

"Uninstalling or modifying IBM MQ on HP-UX" on page 296
On HP-UX, you can uninstall the IBM MQ server or client using the **swremove** command. You can also modify an IBM MQ installation by uninstalling selected IBM MQ components.

Related information:

setmqinst

Changing the primary installation

Installing the IBM MQ server silently on HP-UX

▶ HP-UX

You can perform a non-interactive installation of the IBM MQ server using the **swinstall** command. A non-interactive installation is also known as a silent, or unattended installation.

Before you begin

Before you start the installation procedure, make sure that you have completed the necessary steps outlined in "Preparing the system on HP-UX" on page 268.

About this task

This topic describes the non-interactive installation of a server, using the **swinstall** program to select which components you want to install. The components and are listed in "IBM MQ components and features" on page 192.

Procedure

1. Log in as root, or switch to the superuser using the **su** command.
2. Set your current directory to the location of the installation file. The location might be the mount point of the CD, a network location, or a local file system directory.
3. Accept the IBM MQ license agreement without an interactive prompt by entering the following command:
   ```
   ./mqlicense.sh -accept
   ```
4. Install IBM MQ using the **swinstall** command:
   a. If this installation is not the first installation on the system, you must add `-x allow_multiple_versions=true` to the **swinstall** command.
   b. Add the names of the components to install as parameters of the **swinstall** command. The installer automatically resolves any dependencies.
   c. Optional: Identify the installation location by adding **,l=** *MQ_INSTALLATION_PATH* as a parameter of the **swinstall** command. For each installation, all of the IBM MQ components that you require must be installed in the same location. The installation path specified must either be an empty directory, the root of an unused file system, or a path that does not exist. The length of the path is limited to 256 bytes and must not contain spaces.

For example, to install all IBM MQ components, in a non-default location, as the first installation, enter the following command:

```
swinstall -s /installation_file.v11 MQSERIES,l=/opt/customLocation
```

To perform a partial installation, providing a list of components, in the default location, as the second installation, enter the following command:

```
swinstall -x allow_multiple_versions=true -s /installation_file.v11
MQSERIES.MQM-RUNTIME MQSERIES.MQM-BASE MQSERIES.MQM-SERVER
```

`/installation_file.v11` is the absolute path to the installation file. The path must begin with a `/` and end with the name of the installation file. The installation file has the extension `.v11`.

5. If this installation is not the first installation on the system, you must enter the following command to configure the installation:

   Note: *MQ_INSTALLATION_PATH* is the path where you have just installed IBM MQ and the character defining the path is a lower case `L`.

   ```
   swconfig -x allow_multiple_versions=true MQSERIES,l=MQ_INSTALLATION_PATH
   ```

   If you do not enter this command, the **swlist** command reports the installation as installed instead of configured. You must not use IBM MQ unless the installation is configured.

Example

The example shows the command to run a silent, full installation in the default location, using the alternative form of specifying the source depot using `-x source_directory=` instead of `-s`. Notice that all the language features are installed. Run a partial installation to install your chosen languages.

```
cd /downloads/WMQInstallFiles
swinstall -v -x source_directory=$PWD/hpUxxxxx.v11 MQSERIES
```

What to do next

- If you have chosen this installation to be the primary installation on the system, you must now set it as the primary installation. Enter the following command at the command prompt:

  ```
  MQ_INSTALLATION_PATH/bin/setmqinst -i -p MQ_INSTALLATION_PATH
  ```

  where *MQ_INSTALLATION_PATH* represents the directory where IBM MQ is installed.

  You can have only one primary installation on a system. If there is already a primary installation on the system, you must unset it before you can set another installation as the primary installation. For more information, see Changing the primary installation.
- You might want to set up the environment to work with this installation. You can use the **setmqenv** or **crtmqenv** command to set various environment variables for a particular installation of IBM MQ . For more information, see setmqenv and crtmqenv.
- If you want to confirm that the installation was successful, you can verify your installation. See "Verifying an IBM MQ installation on HP-UX" on page 282, for more information.

Related concepts:

"Multiple installations on UNIX, Linux, and Windows" on page 200
On UNIX, Linux, and Windows, it is possible to have more than one copy of IBM MQ on a system.

"Primary installation on UNIX, Linux, and Windows" on page 202
On systems that support multiple installations of IBM MQ (UNIX, Linux, and Windows), the primary installation is the one to which IBM MQ system-wide locations refer. Having a primary installation is optional, but convenient.

Related tasks:

"Installing IBM MQ server on HP-UX" on page 274
You can install an IBM MQ server on HP-UX either interactively or silently.

"Uninstalling or modifying IBM MQ on HP-UX" on page 296
On HP-UX, you can uninstall the IBM MQ server or client using the **swremove** command. You can also modify an IBM MQ installation by uninstalling selected IBM MQ components.

Related information:

setmqinst

Changing the primary installation

Converting a trial license on HP-UX

HP-UX

Convert a trial license to a full license without reinstalling IBM MQ.

When the trial license expires, the "count-down" displayed by the **strmqm** command informs you the license has expired, and the command does not run.

Before you begin

1. IBM MQ is installed with a trial license.
2. You have access to the installation media of a fully licensed copy of IBM MQ.

About this task

Run the **setmqprd** command to convert a trial license to a full license.

If you do not want to apply a full license to your trial copy of IBM MQ, you can uninstall it at any time.

Procedure

1. Obtain the full license from the fully licensed installation media.

   The full license file is `amqpcert.lic`. On HP-UX, it is in the */MediaRoot/*`licenses` directory on the installation media.
2. Run the **setmqprd** command from the installation that you are upgrading:
   ```
   MQ_INSTALLATION_PATH/bin/setmqprd /MediaRoot/licenses/amqpcert.lic
   ```

Related information:
setmqprd

Displaying messages in your national language on HP-UX

HP-UX

To display messages from a different national language message catalog, you must install the appropriate catalog and set the **LANG** environment variable.

About this task

Messages in U.S. English are automatically installed with IBM MQ

Message catalogs for all languages are installed in *MQ_INSTALLATION_PATH*/msg/*language identifier*, where *language identifier* is one of the identifiers in Table 41.

If you require messages in a different language, perform the following steps:

Procedure

1. Install the appropriate message catalog (see "IBM MQ components and features" on page 192).
2. To select messages in a different language, ensure the **LANG** environment variable is set to the identifier for the language you want to install:

Table 41. Language identifiers

Identifier	**Language**
cs_CZ	Czech
de_DE	German
es_ES	Spanish
fr_FR	French
hu_HU	Hungarian
it_IT	Italian
ja_JP	Japanese
ko_KR	Korean
pl_PL	Polish
pt_BR	Brazilian Portuguese
ru_RU	Russian
zh_CN	Simplified Chinese
zh_TW	Traditional Chinese

Installing an IBM MQ client on HP-UX

HP-UX

You can interactively install the IBM MQ client for HP-UX using `swinstall`.

Before you begin

Before you start the installation procedure, make sure that you have completed the necessary steps outlined in “Preparing the system on HP-UX” on page 268.

About this task

This topic describes the installation of a client, using the `swinstall` program to select which components you want to install. The components and are listed in “IBM MQ components for HP-UX” on page 264; you must install at least the Runtime and Client components.

Procedure

1. Log in as root, or switch to the superuser using the **su** command.
2. Make your current directory the location of the installation file. The location might be the mount point of the DVD, a network location, or a local file system directory.
3. Accept the license by running the `mqlicense` script:
   ```
   ./mqlicense.sh
   ```
   The license is displayed. If you accept the license, you can continue the installation.
4. Type the following command to start the interactive installation procedure:
   ```
   swinstall -s installation_file
   ```
   installation_file is the absolute path to the installation file. The path must begin with a / and end with the name of the installation file. The installation file has the extension `.v11`.

   If the files on your DVD are in uppercase with a ";1" suffix, use this name for the depot.
5. In the resulting menu screen, select **MQSERIES**.
   a. If you do not want to install all IBM MQ components, open **MQSERIES**
      1) Mark the components you want to install. The installer resolves dependencies automatically.
      2) Review the information displayed by the installer.
6. Optional: To install IBM MQ to a non-default location, select **Actions > Change Product Location**.

   For each installation, all of the IBM MQ components that you require must be installed in the same location.

   The installation path specified must either be an empty directory, the root of an unused file system, or a path that does not exist. The length of the path is limited to 256 bytes and must not contain spaces.
7. Select **Actions > Install**. The log file tells you if there are any problems that need fixing.
8. Fix any problems, and click **OK** to install. You are informed when the installation has finished.
9. If this installation is not the first installation on the system, you must enter the following command:
   ```
   swconfig -x allow_multiple_versions=true MQSERIES,l= MQ_INSTALLATION_PATH
   ```
   where *MQ_INSTALLATION_PATH* is the path where you have just installed IBM MQ. If you do not enter this command, the **swlist** command reports the installation as installed instead of configured. You must not use IBM MQ unless the installation is configured.

What to do next

- If you have chosen this installation to be the primary installation on the system, you must now set it as the primary installation. Enter the following command at the command prompt:

```
MQ_INSTALLATION_PATH/bin/setmqinst -i -p MQ_INSTALLATION_PATH
```

You can have only one primary installation on a system. If there is already a primary installation on the system, you must unset it before you can set another installation as the primary installation. For more information, see Changing the primary installation.

- You might want to set up the environment to work with this installation. You can use the **setmqenv** or **crtmqenv** command to set various environment variables for a particular installation of IBM MQ. For more information, see setmqenv and crtmqenv.
- For instructions on how to verify your installation, see "Testing communication between a client and a server on HP-UX" on page 295.

Related tasks:

"Uninstalling or modifying IBM MQ on HP-UX" on page 296
On HP-UX, you can uninstall the IBM MQ server or client using the **swremove** command. You can also modify an IBM MQ installation by uninstalling selected IBM MQ components.

Installing an IBM MQ client silently on HP-UX

HP-UX

You can perform a non-interactive installation of the IBM MQ client using the **swinstall** command. A non-interactive installation is also known as a silent, or unattended installation.

Before you begin

Before you start the installation procedure, make sure that you have completed the necessary steps outlined in "Preparing the system on HP-UX" on page 268.

About this task

This topic describes the non-interactive installation of a client, using the **swinstall** program to select which components you want to install. The components and are listed in "IBM MQ components and features" on page 192 ; you must install at least the Runtime and client components.

Procedure

1. Log in as root, or switch to the superuser using the **su** command.
2. Make your current directory the location of the installation file. The location might be the mount point of the CD, a network location, or a local file system directory.
3. Accept the IBM MQ license agreement without an interactive prompt by entering the following command:
   ```
   ./mqlicense.sh -accept
   ```
4. Install IBM MQ using the **swinstall** command:
   a. If this installation is not the first installation on the system, you must add `-x allow_multiple_versions=true` to the **swinstall** command.
   b. Add the names of the components to install as parameters of the **swinstall** command. The installer automatically resolves any dependencies.
   c. Optional: Identify the installation location by adding `,l=` *MQ_INSTALLATION_PATH* as a parameter of the **swinstall** command. For each installation, all of the IBM MQ components that you require must be installed in the same location. The installation path specified must either be an empty directory, the root of an unused file system, or a path that does not exist. The length of the path is limited to 256 bytes and must not contain spaces.

   For example, to install all IBM MQ components, in a non-default location, as the first installation, enter the following command:
   ```
   swinstall -s /installation_file.v11 MQSERIES,l=/opt/customLocation
   ```

To perform a partial installation, providing a list of components, in the default location, as the second installation, enter the following command:

```
swinstall -s /installation_file.v11
MQSERIES.MQM-RUNTIME MQSERIES.MQM-BASE MQSERIES.MQM-CL-HPUX -x allow_multiple_versions=true
```

`/installation_file.v11` is the absolute path to the installation file. The path must begin with a `/` and end with the name of the installation file. The installation file has the extension `.v11`.

5. If this installation is not the first installation on the system, you must enter the following command:

```
swconfig -x allow_multiple_versions=true MQSERIES,l= MQ_INSTALLATION_PATH
```

where *MQ_INSTALLATION_PATH* is the path where you have just installed IBM MQ. If you do not enter this command, the **swlist** command reports the installation as installed instead of configured. You must not use IBM MQ unless the installation is configured.

What to do next

For instructions on how to verify your installation, see "Testing communication between a client and a server on HP-UX" on page 295.

Verifying an IBM MQ installation on HP-UX

HP-UX

The topics in this section provide instructions on how to verify a server or a client installation of IBM MQ on HP-UX systems.

About this task

You can verify a local (stand-alone) server installation or a server-to-server installation of the IBM MQ server:
- A local server installation has no communication links with other IBM MQ installations.
- A server-to-server installation does have links to other installations.

You can also verify that your IBM MQ MQI client installation completed successfully and that the communication link is working.

Procedure

- To verify a local server installation, see "Verifying a local server installation on HP-UX" on page 283.
- To verify a server-to-server installation, see "Verifying a server-to-server installation on HP-UX" on page 286.
- To verify a client installation, see "Verifying a client installation using the command line on HP-UX" on page 292.

Verifying a local server installation on HP-UX

HP-UX

You can use either the command line or the postcard application to verify a local (stand-alone) installation on HP-UX.

About this task

You can use the command line to verify that IBM MQ is successfully installed, and that the associated communication links are working properly.

You can also verify an installation using the postcard application. The postcard application is Java based and requires a system with the ability to view a graphical display.

Procedure

- To use the command line to verify an installation, see "Verifying a local server installation using the command line on HP-UX."
- To use the postcard application to verify an installation, see "Verifying a local server installation using the Postcard application on HP-UX" on page 285.

Verifying a local server installation using the command line on HP-UX: HP-UX

On HP-UX systems, you can verify a local installation by using the command line to create a simple configuration of one queue manager and one queue. You can also verify an installation using the postcard application.

Before you begin

To verify the installation, you must first install the samples package.

Before beginning the verification procedure, you might want to check that you have the latest fixes for your system. For more information about where to find the latest updates, see "Checking requirements on Windows" on page 444.

About this task

Use the following steps to configure your default queue manager from the command line. After the queue manager is configured, use the `amqsput` sample program to put a message on the queue. You then use the `amqsget` sample program to get the message back from the queue.

IBM MQ object definitions are case-sensitive. Any text entered as an MQSC command in lowercase is converted automatically to uppercase unless you enclose it in single quotation marks. Make sure that you type the examples exactly as shown.

Procedure

1. On an HP-UX system, log in as a user in the `mqm` group.
2. Set up your environment:
   a. Set up environment variables for use with a particular installation by entering one the following command:
   ```
   . MQ_INSTALLATION_PATH/bin/setmqenv -s
   ```
   where *`MQ_INSTALLATION_PATH`* refers to the location where IBM MQ is installed.
   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 a queue manager called `QMA` by entering the following command:

   ```
   crtmqm QMA
   ```

   Messages indicate when the queue manager is created, and when the default IBM MQ objects are created.
4. Start the queue manager by entering the following command:

   ```
   strmqm QMA
   ```

   A message indicates when the queue manager starts.
5. Start MQSC by entering the following command:

   ```
   runmqsc QMA
   ```

   A message indicates when MQSC starts. MQSC has no command prompt.
6. Define a local queue called `QUEUE1` by entering the following command:

   ```
   DEFINE QLOCAL (QUEUE1)
   ```

   A message indicates when the queue is created.
7. Stop MQSC by entering the following command:

   ```
   end
   ```

   Messages are shown, followed by the command prompt.

Note: Subsequent steps require that the samples package is installed.

8. Change into the *`MQ_INSTALLATION_PATH`*`/samp/bin` directory, which contains the sample programs. *`MQ_INSTALLATION_PATH`* represents the high-level directory in which IBM MQ is installed.
9. Put a message on the queue by entering the following commands

   ```
   ./amqsput QUEUE1 QMA
   ```

   The following messages are shown:

   ```
   Sample AMQSPUT0 start
   target queue is QUEUE1
   ```
10. Type some message text on one or more lines, where each line is a different message. 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. Get the messages from the queue, by entering the following command:

    ```
    ./amqsget QUEUE1 QMA
    ```

    The sample program starts, and your messages are displayed.

Results

You have successfully verified your local installation.

Verifying a local server installation using the Postcard application on HP-UX: HP-UX

Sending messages successfully between two Postcard applications verifies a local installation.

Before you begin

The postcard application is Java based and requires a system with the ability to view a graphical display.

You must ensure that you are a member of the IBM MQ administrators group (`mqm`).

Note: Using Postcard to verify an IBM MQ installation is only possible if there is one IBM MQ installation on that box. The Default Configuration wizard will not create a default configuration if a queue manager already exists on the box. The Default Configuration wizard will run on any installation on a box but only one default configuration can be created per box. Using Postcard to verify second and subsequent installations of IBM MQ on the same box is not possible.

To verify that the local installation is working, you can run two instances of the Postcard application on the same server. The postcard application can send messages to, and receive messages from, other postcard applications. Successful sending and receiving of messages verifies that IBM MQ is installed and working correctly on the server.

Procedure

1. Log on as a user in group `mqm`.
2. Start the postcard application in one of the following ways:
   a. From the command line:
      1) Change the directory to *`MQ_INSTALLATION_PATH`*`/java/bin`. *`MQ_INSTALLATION_PATH`* represents the high-level directory in which IBM MQ is installed.
      2) Run the postcard application by entering the following command:
      ```
      ./postcard
      ```
   b. From the IBM MQ Explorer:
      1) If the Welcome to IBM MQ Explorer Content view page does not show, click **IBM MQ** in the Navigator view to show the Welcome page.
      2) Click **Launch Postcard** to start the Postcard.
3. At the Postcard - Sign On window, type in a nickname to use to send messages within the Postcard application (for example, `User1`).
4. Select the queue manager to use as the mailbox:
   - If you do not have any queue managers, you are prompted to either launch the Default Configuration or close the Postcard application. Launching the Default Configuration creates a default queue manager.
   - If the only queue manager on your server is the default queue manager, this queue manager is used automatically for the postcard application. The default queue manager is created by running the Default Configuration wizard
   - If you have created your own queue managers, but you have not run the Default Configuration wizard, select an appropriate queue manager from the list.
   - If you have run the Default Configuration wizard and you want to use the default queue manager, but there are other queue managers on your server, select the **Advanced** check box. Then select **Use Default Configuration as mailbox**.
   - If you have run the Default Configuration wizard and also created your own queue managers, and you do not want to use the default queue manager, select the **Advanced** check box. Then select **Choose queue manager as mailbox**, and then select the appropriate queue manager from the list.

When your selection is complete, click **OK** to display your first Postcard window.

5. Run a second instance of the Postcard application by following the steps used to open the first instance of the Postcard application.
6. The Postcard - Sign On panel is displayed again. Type in a second nickname to use to send messages within this second Postcard application (for example, `User2`).
7. Repeat the selection of the queue manager that you want to use as the mailbox (as described in step 4). The queue manager you select for this second Postcard must be the same queue manager as used for the first instance of the Postcard application.
8. In the first Postcard, (User1), enter the nickname (`User2`) for the second Postcard application in the **To:** field. Because the sender and receiver are on the same server, you can leave the **On:** field blank.
9. Type a message in the **Message:** field and click **Send**.
10. The **Postcards sent and received** area of the Postcard shows details of the message. In the sending Postcard, the message is displayed as sent. In the receiving Postcard, the message is displayed as received.
11. In the receiving Postcard, (User2), double-click the message in the **Postcards sent and received** area to view it. When this message arrives, it verifies that IBM MQ is correctly installed.

What to do next

Depending on your situation, you might want to do the following tasks:

- Install IBM MQ on other servers. Follow the installation procedure for the appropriate platform. Ensure that you use the **Join Default Cluster** window in the Default Configuration wizard to add the other servers to the cluster on your first server.
- Install the IBM MQ MQI client on other servers.
- Continue with further administration tasks, see Administering IBM MQ.

Verifying a server-to-server installation on HP-UX

HP-UX

You can use either the command line or the postcard application to verify a server-to-server installation on HP-UX.

Before you begin

For a server-to-server verification, the communication links between the two systems must be checked. Before you can do the verification, you must therefore ensure that the communications protocol is installed and configured on both systems.

On HP-UX, IBM MQ supports both TCP and SNA.

The examples in this task use TCP/IP. If you do not use TCP, see Setting up communication on UNIX and Linux.

About this task

For a server-to server installation, you can use the command line to verify that IBM MQ is successfully installed, and that the associated communication links are working properly.

You can also verify an installation using the postcard application. The postcard application is Java based and requires a system with the ability to view a graphical display.

Procedure

- To use the command line to verify an installation, see "Verifying a server-to-server installation using the command line on HP-UX."
- To use the postcard application to verify an installation, see "Verifying a server-to-server installation using the Postcard application on HP-UX" on page 290.

Verifying a server-to-server installation using the command line on HP-UX: HP-UX

You can verify a server-to-server installation using two servers, one as a sender and one as a receiver.

Before you begin

- Make sure that TCP/IP and IBM MQ are installed on both servers (see "Verifying a server-to-server installation on HP-UX" on page 286).
- Make sure that you are a member of the IBM MQ administrators group (**mqm**) on each server.
- Decide which installation is the sender server and which installation is the receiver server. The installations might be on the same system, or on different systems.

About this task

IBM MQ object definitions are case-sensitive. Any text entered as an MQSC command in lowercase is converted automatically to uppercase unless you enclose it in single quotation marks. Make sure that you type the examples exactly as shown.

Procedure

1. On the **receiver** server:
   a. On HP-UX, log in as a user in the mqm group.
   b. Check which ports are free, for example by running **netstat**. For more information about this command, see the documentation of your operating system.

      If port 1414 is not in use, make a note of 1414 to use as the port number in step 2 h. Use the same number for the port for your listener later in the verification. If it is in use, note a port that is not in use; for example 1415.
   c. Set up the environment for the installation you are using by entering the following command at the command prompt:
      ```
      . MQ_INSTALLATION_PATH/bin/setmqenv -s
      ```
      where *MQ_INSTALLATION_PATH* refers to the location where IBM MQ is installed.
   d. Create a queue manager called QMB by entering the following command at the command prompt:
      ```
      crtmqm QMB
      ```
      Messages tell you that the queue manager has been created, and that the default IBM MQ objects have been created.
   e. Start the queue manager by entering the following command:
      ```
      strmqm QMB
      ```
      A message tells you when the queue manager has started.
   f. Start MQSC by entering the following command:
      ```
      runmqsc QMB
      ```
      A message tells you that MQSC has started. MQSC has no command prompt.
   g. Define a local queue called RECEIVER.Q by entering the following command:
      ```
      DEFINE QLOCAL (RECEIVER.Q)
      ```

A message tells you the queue has been created.

h. Define a listener by entering the following command:

```
DEFINE LISTENER (LISTENER1) TRPTYPE (TCP) CONTROL (QMGR) PORT ( PORT_NUMBER )
```

Where *port_number* is the name of the port the listener runs on. This number must be the same as the number used when defining your sender channel.

i. Start the listener by entering the following command:

```
START LISTENER (LISTENER1)
```

Note: Do not start the listener in the background from any shell that automatically lowers the priority of background processes.

j. Define a receiver channel by entering the following command:

```
DEFINE CHANNEL (QMA.QMB) CHLTYPE (RCVR) TRPTYPE (TCP)
```

A message tells you when the channel has been created.

k. End MQSC by typing:

```
end
```

Some messages are displayed, followed by the command prompt.

2. On the **sender** server:

a. As the sender server is an AIX system, log in as a user in the mqm group.

b. Set up the environment for the installation you are using by entering the following command at the command prompt:

```
. MQ_INSTALLATION_PATH/bin/setmqenv -s
```

where *MQ_INSTALLATION_PATH* refers to the location where IBM MQ is installed.

c. Create a queue manager called QMA by entering the following command at the command prompt:

```
crtmqm QMA
```

Messages tell you that the queue manager has been created, and that the default IBM MQ objects have been created.

d. Start the queue manager, by entering the following command:

```
strmqm QMA
```

A message tells you when the queue manager has started.

e. Start MQSC by entering the following command:

```
runmqsc QMA
```

A message tells you that an MQSC session has started. MQSC had no command prompt.

f. Define a local queue called QMB (to be used as a transmission queue) by entering the following command:

```
DEFINE QLOCAL (QMB) USAGE (XMITQ)
```

A message tells you when the queue has been created.

g. Define a local definition of the remote queue with by entering the following command:

```
DEFINE QREMOTE (LOCAL.DEF.OF.REMOTE.QUEUE) RNAME (RECEIVER.Q) RQMNAME ('QMB') XMITQ (QMB)
```

h. Define a sender channel by entering one of the following commands:

con-name is the TCP/IP address of the receiver system. If both installations are on the same system, the *con-name* is localhost. *port* is the port you noted in 1 b. If you do not specify a port, the default value of 1414 is used.

```
DEFINE CHANNEL (QMA.QMB) CHLTYPE (SDR) CONNAME ('CON-NAME(PORT)') XMITQ (QMB) TRPTYPE (TCP)
```

i. Start the sender channel by entering the following command:

```
START CHANNEL(QMA.QMB)
```

The receiver channel on the receiver server starts automatically when the sender channel starts.

j. Stop MQSC by entering the following command:

```
end
```

Some messages are displayed, followed by the command prompt.

k. If the sender server is a UNIX or Linux system, change into the *MQ_INSTALLATION_PATH*/samp/bin directory. This directory contains the sample programs. *MQ_INSTALLATION_PATH* represents the high-level directory in which IBM MQ is installed.

l. If both the sender server and receiver server are installations on the same system, check that the queue managers have been created on different installations by entering the following command:

```
dspmq -o installation
```

If the queue managers are on the same installation, move either QMA to the sender installation or QMB to the receiver installation by using the **setmqm** command. For more information, see setmqm.

m. Put a message on the local definition of the remote queue, which in turn specifies the name of the remote queue. Enter one of the following commands:
   - On Windows:

     ```
     amqsput LOCAL.DEF.OF.REMOTE.QUEUE QMA
     ```
   - On UNIX and Linux:

     ```
     ./amqsput LOCAL.DEF.OF.REMOTE.QUEUE QMA
     ```

   A message tells you that amqsput has started.

n. Type some message text on one or more lines, followed by a blank line. A message tells you that amqsput has ended. Your message is now on the queue and the command prompt is displayed again.

3. On the **receiver** server:

   a. As your receiver server is an AIX system, change into the *MQ_INSTALLATION_PATH*/samp/bin directory. This directory contains the sample programs. *MQ_INSTALLATION_PATH* represents the high-level directory in which IBM MQ is installed.

   b. Get the message from the queue on the receiver by entering the following command:

   ```
   ./amqsget RECEIVER.Q QMB
   ```

   The sample program starts, and your message is displayed. After a pause, the sample ends. Then the command prompt is displayed.

Results

You have now successfully verified the server-to-server installation.

Verifying a server-to-server installation using the Postcard application on HP-UX: HP-UX

You can use two instances of the Postcard application to verify that a server-to-server installation is working.

Before you begin

You can use the Postcard application on two servers, one instance of the Postcard application on each server, to verify that a server-to-server installation is working. Successful sending and receiving of messages verifies that IBM MQ is successfully installed, and that communication between the two servers is working correctly.

Note:

- If the system has multiple IBM MQ installations, ensure that Postcard has not been run before on any installations on that server. As the default configuration can only exist on one IBM MQ installation per system, the Default Configuration wizard and Postcard can not be used for verification of a second or any subsequent installation.
- The two server installations must be on different systems to do a server-to-server verification using the postcard application. To verify a server-to-server installation on the same machine, you can use the command line.
- Make sure that TCP/IP and IBM MQ are installed on both servers.
- Make sure that your systems are able to view a graphical display.
- Make sure that you are a member of the IBM MQ administrators group (**mqm**) on each server.
- Check that one of the following scenarios applies:
  - Neither server has had any queue managers created.
  - Use the Default Configuration wizard to create default queue managers on each server and link them to the default cluster.

    Details on how to use the Default Configuration wizard are provided in this topic.
  - Both servers have existing queue managers and these queue managers are in the same cluster.

    If your queue managers are not in the same cluster, create new queue managers on both servers. Then create a cluster, and ensure that the queue managers that you create on each server belong to that cluster.
  - You have configured channels to communicate between the two servers.

    For instructions on how to set up the channels, see "Verifying a server-to-server installation using the command line on HP-UX" on page 287. After you have set up the channels, follow the instructions in this topic to verify your server-to-server installation.

Procedure

1. On the first server, log on as a user in group **mqm**.
2. Start the postcard application in one of the following ways:
   a. From the command line:
      1) Change the directory to *MQ_INSTALLATION_PATH*/java/bin. *MQ_INSTALLATION_PATH* represents the high-level directory in which IBM MQ is installed.
      2) Run the postcard application by entering the following command:

```
./postcard
```

   b. From the IBM MQ Explorer:
      1) If the Welcome to IBM MQ Explorer Content view page does not show, click **IBM MQ** in the Navigator view to show the Welcome page.
      2) Click **Launch Postcard** to start the Postcard.

3. At the Postcard - Sign On window, type a nickname to use to send messages within the Postcard application. For example, `User1` for the first server, and `User2` for the second server.
4. When you have completed the wizard, you are taken back to the Postcard - Sign On window.
5. Select the queue manager to use as the mailbox:
   - If you do not have any queue managers, you are prompted to either launch the Default Configuration or close the Postcard application. Work through the Default Configuration wizard. When you get to the option to join the queue manager to the default cluster, tick the check box. On the next screen:
     – For the first server, select **yes, make it the repository for the cluster**.
     – For the second server, select **No another computer has already joined the cluster as a repository**. When requested, enter the location of the repository, by typing the name of the sender server.
   - If the only queue manager on your server is the default queue manager, this queue manager is used automatically for the postcard application. The default queue manager is created by running the Default Configuration wizard
   - If you have created your own queue managers, but you have not run the Default Configuration wizard, select an appropriate queue manager from the list.
   - If you have run the Default Configuration wizard and you want to use the default queue manager, but there are other queue managers on your server, select the **Advanced** check box. Then select **Use Default Configuration as mailbox**.
   - If you have run the Default Configuration wizard and also created your own queue managers, and you do not want to use the default queue manager, select the **Advanced** check box. Then select **Choose queue manager as mailbox**, and then select the appropriate queue manager from the list.

   When your selection is complete, click **OK**.
6. Complete steps 1 - 5 for the second server.
7. In the Postcard on the first server:
   a. Enter the nickname (`user2`) for the Postcard application on the second server in the **To:** field.
   b. Enter the queue manager on the second server in the **On:** field.
   c. Type a message in the **Message:** field and click **Send**.
8. In the Postcard on the second server:
   a. In the **Postcards sent and received**, double-click the message marked as received to view the message from the first server.
   b. Optional: Send a postcard to the first server by adapting the instructions in step 7. You must enter details of the first server in the **To:** field and the **On:** field.

   The messages verify that IBM MQ is correctly installed and that your communication link between the two servers is working correctly.

Verifying a client installation using the command line on HP-UX

HP-UX

You can verify a client installation using the command line. On the server you create a queue manager, a local queue, a listener, and a server-connection channel. You must also apply security rules to allow the client to connect and make use of the queue defined. On the client you create a client-connection channel, and then use the sample PUT and GET programs to complete the verification procedure.

The verification procedure shows how to create a queue manager called `queue.manager.1`, a local queue called `QUEUE1`, and a server-connection channel called `CHANNEL1` on the server.

It shows how to create the client-connection channel on the IBM MQ MQI client workstation. It then shows how to use the sample programs to put a message onto a queue, and get the message from the queue.

The example does not address any client security issues. See Setting up IBM MQ MQI client security for details if you are concerned with IBM MQ MQI client security issues.

The verification procedure assumes that:
- The full IBM MQ server product has been installed on a server.
- The server installation is accessible on your network.
- The IBM MQ MQI client software has been installed on a client system.
- The IBM MQ sample programs have been installed.
- TCP/IP has been configured on the server and client systems. For more information, see Configuring connections between the server and client.

First, set up the server using the command line, using the instructions in “Setting up the server using the command line on HP-UX.”

Once you have set up the server, you must set up the client, using the instructions in “Connecting to a queue manager, using the `MQSERVER` environment variable on HP-UX” on page 294.

Finally, you can test the communications between client and server, using the instructions in “Testing communication between a client and a server on HP-UX” on page 295.

Setting up the server using the command line on HP-UX: HP-UX

Follow these instructions to create a queue manager, queue, and channel on the server. You can then use these objects to verify the installation.

About this task

These instructions assume that no queue manager or other IBM MQ objects have been defined.

IBM MQ object definitions are case-sensitive. Any text entered as an MQSC command in lowercase is converted automatically to uppercase unless you enclose it in single quotation marks. Make sure that you type the examples exactly as shown.

Procedure

1. Create a user ID on the server that is not in the `mqm` group. This user ID must exist on the server and client. This is the user ID that the sample applications must be run as, otherwise a 2035 error is returned.
2. Log in as a user in the mqm group.

3. You must set various environment variables so that the installation can be used in the current shell. You can set the environment variables by entering the following command:

```
. MQ_INSTALLATION_PATH/bin/setmqenv -s
```

   where `MQ_INSTALLATION_PATH` refers to the location where IBM MQ is installed.
4. Create a queue manager called `QUEUE.MANAGER.1` by entering the following command:

   ```
   crtmqm QUEUE.MANAGER.1
   ```

   You see messages telling you that the queue manager has been created.
5. Start the queue manager by entering the following command:

   ```
   strmqm QUEUE.MANAGER.1
   ```

   A message tells you when the queue manager has started.
6. Start MQSC by entering the following command:

   ```
   runmqsc QUEUE.MANAGER.1
   ```

   A message tells you that an MQSC session has started. MQSC has no command prompt.
7. Define a local queue called `QUEUE1` by entering the following command:

   ```
   DEFINE QLOCAL(QUEUE1)
   ```

   A message tells you when the queue has been created.
8. Allow the user ID that you created in step 1 to use `QUEUE1` by entering the following command:

   ```
   SET AUTHREC PROFILE(QUEUE1) OBJTYPE(QUEUE) PRINCIPAL(' non_mqm_user ') AUTHADD(PUT,GET)
   ```

   where *non_mqm_user* is the user ID created in step 1. A message tells you when the authorization has been set. You must also run the following command to give the user ID authority to connect:

   ```
   SET AUTHREC OBJTYPE(QMGR) PRINCIPAL(' non_mqm_user ') AUTHADD(CONNECT)
   ```

   If this command is not run, a 2305 stop error is returned.
9. Define a server-connection channel by entering the following command:

   ```
   DEFINE CHANNEL (CHANNEL1) CHLTYPE (SVRCONN) TRPTYPE (TCP)
   ```

   A message tells you when the channel has been created.
10. Allow your client channel to connect to the queue manager and run under the user ID that you created in step 1, by entering the following MQSC command:

    ```
    SET CHLAUTH(CHANNEL1) TYPE(ADDRESSMAP) ADDRESS(' client_ipaddr ') MCAUSER(' non_mqm_user ')
    ```

    where *client_ipaddr* is the IP address of the client system, and *non_mqm_user* is the user ID created in step 1. A message tells you when the rule has been set.
11. Define a listener by entering the following command:

    ```
    DEFINE LISTENER (LISTENER1) TRPTYPE (TCP) CONTROL (QMGR) PORT (port_number)
    ```

    where *port_number* is the number of the port the listener is to run on. This number must be the same as the number used when defining your client-connection channel in "Installing an IBM MQ client on HP-UX" on page 280.

    Note: If you omit the port parameter from the command, a default value of `1414` is used for the listener port. If you want to specify a port other than `1414`, you must include the port parameter in the command, as shown.
12. Start the listener by entering the following command:

    ```
    START LISTENER (LISTENER1)
    ```
13. Stop MQSC by entering:

```
end
```

You see some messages, followed by the command prompt.

What to do next

Follow the instructions to set up the client. See "Connecting to a queue manager, using the `MQSERVER` environment variable on HP-UX."

Connecting to a queue manager, using the `MQSERVER` environment variable on HP-UX: HP-UX

When an IBM MQ application is run on the IBM MQ MQI client, it requires the name of the MQI channel, the communication type, and the address of the server to be used. Provide these parameters by defining the `MQSERVER` environment variable.

Before you begin

Before you start this task, you must complete the task, "Setting up the server using the command line on HP-UX" on page 292, and save the following information:
- The host name or IP address of the server and port number that you specified when creating the listener.
- The channel name of the server-connection channel.

About this task

This task describes how to connect an IBM MQ MQI client, by defining the `MQSERVER` environment variable on the client.

You can give the client access to the generated client channel definition table, `amqclchl.tab` instead; see Accessing client-connection channel definitions.

Procedure
1. Log in as the userid that you created in Step 1 of "Setting up the server using the command line on HP-UX" on page 292.
2. Check the TCP/IP connection. From the client, enter one of the following commands:
   - `ping server-hostname`
   - `ping n.n.n.n`

     `n.n.n.n` represents the network address. You can set the network address in IPv4 dotted decimal form, for example, `192.0.2.0`. Alternatively, set the address in IPv6 hexadecimal form, for example `2001:0DB8:0204:acff:fe97:2c34:fde0:3485`.

   If the **`ping`** command fails, correct your TCP/IP configuration.
3. Set the `MQSERVER` environment variable. From the client, enter the following command:
   ```
   export MQSERVER=CHANNEL1/TCP/' server-address (port)'
   ```

   Where:
   - *CHANNEL1* is the server-connection channel name.
   - *server-address* is the TCP/IP host name of the server.
   - *port* is the TCP/IP port number the server is listening on.

   If you do not give a port number, IBM MQ uses the one specified in the `qm.ini` file, or the client configuration file. If no value is specified in these files, IBM MQ uses the port number identified in the TCP/IP services file for the service name `MQSeries`. If an `MQSeries` entry in the services file does

not exist, a default value of 1414 is used. It is important that the port number used by the client and the port number used by the server listener program are the same.

What to do next

Use the sample programs to test communication between the client and server; see "Testing communication between a client and a server on HP-UX."

Testing communication between a client and a server on HP-UX: HP-UX

On the IBM MQ MQI client workstation, use the amqsputc sample program to put a message on the queue at the server workstation. Use the amqsgetc sample program to get the message from the queue back to the client.

Before you begin

Complete the previous topics in this section:
- Set up a queue manager, channels, and queue.
- Open a command window.
- Set system environment variables.

About this task

Note that IBM MQ object definitions are case-sensitive. Text entered as an MQSC command in lowercase is converted automatically to uppercase unless you enclose it in single quotation marks. Make sure that you type the examples exactly as shown.

Procedure

1. Change to the *MQ_INSTALLATION_PATH*/samp/bin directory, which contains the sample programs. *MQ_INSTALLATION_PATH* represents the high-level directory in which IBM MQ is installed.
2. You must set certain environment variables so that the installation can be used in the current shell. You can set the environment variables by entering the following command:
   ```
   . MQ_INSTALLATION_PATH/bin/setmqenv -s
   ```
   where *MQ_INSTALLATION_PATH* refers to the location where IBM MQ is installed.
3. Start the PUT program for QUEUE1 on QUEUE.MANAGER.1 by entering the following command:
   ```
   ./amqsputc QUEUE1 QUEUE.MANAGER.1
   ```
   If the command is successful, the following messages are displayed:
   ```
   Sample AMQSPUT0 start target queue is QUEUE1
   ```
   Tip: You might get the error, MQRC_NOT_AUTHORIZED (2035). By default, channel authentication is enabled when a queue manager is created. Channel authentication prevents privileged users accessing a queue manager as an IBM MQ MQI client. For verifying the installation, you can either change the MCA user ID to a non-privileged user, or disable channel authentication. To disable channel authentication run the following MQSC command:
   ```
   ALTER QMGR CHLAUTH(DISABLED)
   ```
   When you finish the test, if you do not delete the queue manager, re-enable channel authentication:
   ```
   ALTER QMGR CHLAUTH(ENABLED)
   ```
4. Type some message text, then press **Enter** twice. The following message is displayed:
   ```
   Sample AMQSPUT0 end
   ```
   Your message is now on the queue that is on the server queue manager.

5. Start the GET program for QUEUE1 on QUEUE.MANAGER.1 by entering the following command:
   ```
   ./amqsgetc QUEUE1 QUEUE.MANAGER.1
   ```
   The sample program starts, and your message is displayed. After a short pause (approximately 30 seconds), the sample ends and the command prompt is displayed again.

Results

You have now successfully verified the client installation.

What to do next

1. You must set various environment variables on the server so that the installation can be used in the current shell. You can set the environment variables by entering the following command:
   ```
   . MQ_INSTALLATION_PATH/bin/setmqenv -s
   ```
   where *MQ_INSTALLATION_PATH* refers to the location where IBM MQ is installed.
2. On the server, stop the queue manager by entering the following command:
   ```
   endmqm QUEUE.MANAGER.1
   ```
3. On the server, delete the queue manager by entering the following command:
   ```
   dltmqm QUEUE.MANAGER.1
   ```

Uninstalling or modifying IBM MQ on HP-UX

HP-UX

On HP-UX, you can uninstall the IBM MQ server or client using the **swremove** command. You can also modify an IBM MQ installation by uninstalling selected IBM MQ components.

Before you begin

If any updates have been applied, remove them before starting the uninstallation procedure. For more information, see Restoring the previous maintenance level on IBM MQ on HP-UX.

Important: You must stop all IBM MQ queue managers, other objects, and applications, before you begin the process to uninstall or modify IBM MQ.

Procedure

1. Stop all IBM MQ applications associated with the installation you are uninstalling or modifying, if you have not already done so.
2. For a server installation, end any IBM MQ activity associated with the installation you are uninstalling or modifying:
   a. Log in as a user in the group mqm.
   b. Set up your environment to work with the installation you want to uninstall or modify. Enter the following command:
      ```
      . MQ_INSTALLATION_PATH/bin/setmqenv
      ```
      where *MQ_INSTALLATION_PATH* refers to the location where IBM MQ is installed.
   c. Display the state of all queue managers on the system. Enter the following command:
      ```
      dspmq -o installation
      ```
   d. Stop all running queue managers associated with the installation you want to uninstall or modify. Enter the following command for each queue manager:
      ```
      endmqm QMgrName
      ```

   e. Stop any listeners associated with the queue managers. Enter the following command for each queue manager:
      ```
      endmqlsr -m QMgrName
      ```
3. Log in as root.
4. Uninstall IBM MQ using **swremove**:
   - To uninstall all IBM MQ components, enter the following command:
     ```
     swremove MQSERIES,l= MQ_INSTALLATION_PATH
     ```
     where *MQ_INSTALLATION_PATH* is the path where IBM MQ is installed.
   - To modify an IBM MQ installation and uninstall selected IBM MQ components, enter the following command:
     ```
     swremove componentname,l= MQ_INSTALLATION_PATH
     ```
     where *componentname* is the name of the component to uninstall, and *MQ_INSTALLATION_PATH* is the path where IBM MQ is installed.

     For example:
     – To uninstall the client component, from an installation in `/opt/myLocation`, enter the following command:
       ```
       swremove MQSERIES.MQM-CL-HPUX,l=/opt/myLocation
       ```
     – To uninstall the client and the telemetry client components, from an installation in `/opt/myLocation`, enter the following command:
       ```
       swremove MQSERIES.MQM-CL-HPUX,l=/opt/myLocation MQSERIES.MQM-TXCLIENT,l=/opt/myLocation
       ```

Results

After uninstallation, certain files under the directory trees `/var/mqm` and `/etc/opt/mqm` are not removed. These files contain user data and remain so subsequent installations can reuse the data. Most of the remaining files contain text, such as INI files, error logs, and FDC files. The directory tree `/var/mqm/shared` contains files that are shared across installations, including the executable shared libraries `libmqzsd.so` and `libmqzsd_r.so`.

What to do next

- If the product successfully uninstalled, you can delete any files and directories contained in the installation directory.
- If there are no other IBM MQ installations on the system, and you are not planning to reinstall or migrate, you can delete the `/var/mqm` and `/etc/opt/mqm` directory trees, including the files `libmqzsd.so` and `libmqzsd_r.so`. Deleting these directories destroys all queue managers and their associated data.

Installing and uninstalling IBM MQ on IBM i

IBM i

Installation tasks that are associated with installing IBM MQ on IBM i systems are grouped in this section.

About this task

To prepare for installation and to install the IBM MQ components, complete the following tasks.

For information about how to uninstall IBM MQ, see "Uninstalling IBM MQ for IBM i" on page 325.

If product fixes or updates are made available, see IBM MQ maintenance tasks for information about how to apply these changes.

Procedure

1. Check the system requirements. See "Hardware and software requirements on IBM i systems" on page 299.
2. Plan your installation.
   - As part of the planning process, you must choose which components to install and where to install them. See "IBM MQ components for IBM i."
   - You must also make some platform-specific choices. See "Planning to install IBM MQ on IBM i" on page 301.
3. Prepare your system for installation of IBM MQ. See "Preparing the system on IBM i" on page 301.
4. Install IBM MQ server. See "Installing IBM MQ server on IBM i" on page 303.
5. Optional: Install an IBM MQ client. See "Installing an IBM MQ client on IBM i" on page 316.
6. Verify your installation. See "Verifying an IBM MQ installation on IBM i" on page 321.

IBM MQ components for IBM i

IBM i

The IBM MQ components that are available for IBM i.

Important: See IBM MQ license information for details of what each purchase of IBM MQ entitles you to install.

The components are as follows:

Server (Base)
Support to enable you to create and support your own applications. This includes the runtime component that provides support for external applications. It also includes support for client connections from IBM MQ installations on other computers.

Command Reference
Help for the CL commands is provided in HTML format and installed with the product in the /QIBM/ProdData/mqm/doc directory.

Samples (Option 1)
Sample application programs. The source is supplied in the QMQMSAMP library and executable files are supplied in the QMQM library.

AMS (Option 2)
The AMS component.

Documentation
The full product documentation is supplied on the IBM MQ Documentation CD.

Readme file
Latest information about the product that became available after publication of this product documentation or the full documentation. You can find the readme file in the root of the product or documentation CD. Review it before starting to install IBM MQ for IBM i.

Managed File Transfer (MFT) components

***BASE**
Support to enable you to create and support your own MFT applications. It also includes support for client connections from IBM MQ MFT installations on other computers.

2 Tools support

3 Agent

4 Services

You must install `*BASE` first because the other three options depend on `*BASE`. Note that option `4` requires that option 3 is installed.

Related concepts:
"IBM MQ components and features" on page 192
You can select the components or features that you require when you install IBM MQ.

Hardware and software requirements on IBM i systems

IBM i

Check that the server environment meets the prerequisites for installing IBM MQ for IBM i. Check the product readme files and install missing prerequisite software supplied on the server CD.

Before installation, you must check that your system meets the hardware and software requirements set out in the IBM MQ system requirements page. See System Requirements for IBM MQ. You must also review the release notes file, which are on the product CD in the `\Readmes` folder for each national language, and check the `READADD.txt` file for any changes made between translation and the manufacturing of the installation CD. `READADD.txt` is found in the root directory of the server installation CD.

During installation, the release notes file is copied to the IBM MQ program files folder (default `/QIBM/ProdData/mqm`).

Storage requirements for IBM MQ server

The storage requirements for IBM i depend on which components you install, and how much working space you need. The storage requirements also depend on the number of queues that you use, the number and size of the messages on the queues, and whether the messages are persistent. You also require archiving capacity on disk, tape, or other media. For more information, see System Requirements for IBM MQ.

Disk storage is also required:
- Prerequisite software
- Optional software
- Your application programs

Installing prerequisite software

To install the prerequisite software provided on the IBM MQ Server CD (which does not include service packs or web browsers), do one of the following:

- Use the IBM MQ installation procedure.

  When you install using the IBM MQ Server CD, there is a **Software Prerequisites** option in the IBM MQ Installation Launchpad window. You can use this option to check what prerequisite software is already installed and which is missing, and to install any missing software.

Using TLS Version 1.2

TLS version 1.2 is the latest version of the Transport Layer Security (TLS) protocol. The core System TLS v1.2 functionality is included in IBM i 7.1 Technology Refresh 6 (TR6). To enable and use the new protocols, program temporary fixes (PTFs) from multiple areas of the operating system are also required.

Provided DCM (5770SS1 option 34) is installed on your system, requesting and applying SI48659 obtains all of the enablement PTFs.

System value changes

The new support is installed, but dormant in System SSL after applying SI48659.

In order to activate the new protocols for System SSL, use Change System Value (CHGSYSVAL) to modify The QSSLPCL system value.

Change the default value of *OPSYS to:

*TLSV1.2

*TLSV1.1

*TLSV1

*SSLV3

If QSSLPCL is set to something other than *OPSYS, add *TLSV1.2 and *TLSV1.1 to the existing setting.

Prerequisite PTFs for multiple certificate support

You are not limited to a single certificate for TLS channels. To use multiple certificates on IBM i platforms, you must install the following program temporary fixes (PTFs):

MF57749

MF57889

SI52214

MF58003

See Digital certificate labels: understanding the requirements for details about how to select certificates by using certificate labels.

Related concepts:

"License requirements" on page 194
You must have purchased sufficient licenses for your installation. The details of the license agreement is stored on your system at installation time so that you can read it at any time. IBM MQ supports IBM License Metric Tool (ILMT).

"Where to find product requirements and support information" on page 195
Before you install IBM MQ, you must check for the latest information and system requirements.

Planning to install IBM MQ on IBM i

IBM i

Before you install IBM MQ on IBM i, you must choose which components to install and where to install them. You must also make some platform-specific choices.

About this task

The following steps provide links to additional information to help you with planning your installation of IBM MQ on IBM i.

Procedure

1. As part of your planning activities, make sure that you review the information on hardware and software requirements for the platform on which you are planning to install IBM MQ. For more information, see "Hardware and software requirements on IBM i systems" on page 299.
2. Decide which IBM MQ components and features to install. See "IBM MQ components and features" on page 192.

   Important: Ensure that your enterprise has the correct license, or licenses, for the components that you are going to install. For more information, see "License requirements" on page 194 and IBM MQ license information.

Preparing the system on IBM i

IBM i

On IBM i systems, you might have to complete several tasks before you install IBM MQ. You might also want to complete other tasks, depending on your installation intentions.

About this task

The tasks that you perform to prepare your systems for installation are listed here. Complete the appropriate tasks for your platform before installing.

Procedure

Configure any additional settings needed for your IBM i system. See "Configuring and tuning the operating system on IBM i" on page 302.

What to do next

When you have completed the tasks to prepare the system, you are ready to start installing IBM MQ. To install a server, see "Installing IBM MQ server on IBM i" on page 303. To install a client, see "Installing an IBM MQ client on IBM i" on page 316.

Related information:
Planning
Maintaining and migrating
IBM MQ maintenance tasks

Configuring and tuning the operating system on IBM i

IBM i

Before installing IBM MQ for IBM i, there are several system values which need to be checked using the DSPSYSVAL command. If necessary, reset the values using the CHGSYSVAL command.

Check the following values and change if required:

QCCSID
Every message has a coded-character set identifier (CCSID) in its header. The CCSID tag identifies the code page and character set of the source.

A queue manager obtains its CCSID from the job that created it. If the job CCSID is not a valid value in the range 1-65534, the queue manager uses the default CCSID value (65535) instead. You can change the CCSID used by the IBM MQ queue manager by using the CL command **CHGMQM**.

Note: The CCSID must be either single-byte character set (SBCS), or mixed, that is SBCS and DBCS. It must not be DBCS only.

QSYSLIBL
Ensure that QSYS2 is included in the list of libraries that make up the system part of the library list. IBM MQ uses programs in this library for data conversion and SNA LU 6.2 communication.

Note: Do not have QMQM as part of the system or user portion of the library list.

QALWOBJRST
Ensure that the QALWOBJRST system value is set to *ALL or *ALWPGMADP before you install MQ. If it is set to *NONE, installation fails.

After installation, reset QALWOBJRST to its original value to maintain system security.

QSHRMEMCTL
Ensure that the QSHRMEMCTL system value is set to 1 (Allowed).

A value of 1 is used in environments where pointers can be shared amongst programs between different jobs.

IBM MQ requires this setting to use the shared memory APIs shmat and shmget and to share its pointers across jobs.

If it is not set correctly, initialization of IBM MQ fails with system return code "3401" (Permission denied), and commands such as CRTMQM, STRMQM, ENDMQM, TRCMQM fail.

QFRCCVNRST
Ensure that the QFRCCVNRST system value is set to 0 (Restore all objects without conversion), or 1 (Objects with validation errors are converted), before you install MQ. If it is not set, installation fails.

QMLTTHDACN
Optionally set this to control the generation of messages into joblogs. Set QMLTTHDACN to 2 to get messages generated in a joblog; set it to 1 to avoid the messages. For example, the message CPD000D is an informational message that is generated when a command that is not thread-safe is issued from a multi-threaded application. Setting QMLTTHDACN to 1 avoids the message.

Related concepts:

"Hardware and software requirements on IBM i systems" on page 299
Check that the server environment meets the prerequisites for installing IBM MQ for IBM i. Check the product readme files and install missing prerequisite software supplied on the server CD.

"License requirements" on page 194
You must have purchased sufficient licenses for your installation. The details of the license agreement is stored on your system at installation time so that you can read it at any time. IBM MQ supports IBM License Metric Tool (ILMT).

Related tasks:

"Installing IBM MQ server on IBM i"
Install IBM MQ for IBM i by installing the IBM MQ server in its primary language, installing samples and installing additional languages.

Installing IBM MQ server on IBM i

IBM i

Install IBM MQ for IBM i by installing the IBM MQ server in its primary language, installing samples and installing additional languages.

Before you begin

Note: Installing the latest version of the IBM MQ server includes client capabilities. Only install the stand-alone client if you do not need the server capabilities.

You have completed planning the installation, obtained the installation CDs and set the system values, see "Configuring and tuning the operating system on IBM i" on page 302. For a complete list of IBM MQ installable services and components for IBM i systems, see Installable services and components for IBM i

About this task

How to install the base IBM MQ server in its primary language, install samples, and install translated versions from a choice of national-languages.

You can install only one instance of IBM MQ for IBM i in each partition of your server.

Procedure

1. Sign on to the system with a user profile that has `*ALLOBJ` special authority, for example `QSECOFR`.
2. Install IBM MQ for IBM i, V9.0 base product, and primary language.

   ```
   RSTLICPGM LICPGM (5724H72) DEV (installation device) OPTION (*BASE) OUTPUT (*PRINT)
   ```

   where the parameters of RSTLICPGM are,

   `LICPGM(5724H72)`
   The product identifier for IBM i.

   `DEV(`*`installation device`*`)`
   The device from which the product is to be loaded, typically an optical drive, for example, `OPT01`.

   `OPTION (*BASE)`
   Install the base IBM MQ for IBM i product.

   `Unspecified parameters`
   Unspecified parameters, such as **`RSTOBJ`** `(*ALL)`, revert to defaults. The command installs both IBM MQ and the language files for the primary language of your system. For installing additional languages, see step 4.

3. Optional: Install the samples using the command:

   ```
   RSTLICPGM LICPGM (5724H72) DEV (installation device) OPTION (1) OUTPUT (*PRINT)
   ```

   Where the parameters of RSTLICPGM are,

   LICPGM (5724H72)
      The product identifier for IBM i.

   DEV (*installation device*)
      The device from which the product is to be loaded, typically an optical drive, for example, `OPT01`.

   OPTION (1)
      Install the samples for IBM i.

   OUTPUT (*PRINT)
      The output is printed with the spooled output of the job.
4. Optional: To install additional languages, sign on to the system with a user profile that has *ALLOBJ special authority. Choose a language code from the table.

Table 42. Globalizations of IBM MQ for IBM i.

Language ID	Language
2909	Belgian English
2966	Belgian French MNCS (Multi-National Character Set)
2980	Brazilian Portuguese
2981	Canadian French MNCS
2975	Czech
2924	English uppercase and lowercase
2984	English US DBCS
2938	English US uppercase DBCS
2928	French
2940	French MNCS
2929	German
2939	German MNCS
2976	Hungarian
2932	Italian
2942	Italian MNCS
2962	Japanese
2930	Japanese Universal
2986	Korean
2978	Polish
2979	Russian
2989	Simplified Chinese
2931	Spanish

- If installing Japanese language feature code 2962, ensure the CCSID of the job installing the product is set to 939 and not 930. Do this to avoid problems with invariant lowercase characters in CCSID 930

  ```
  CHGJOB CCSID(939)
  ```

- If the language feature code is not in the table then the product has not been translated into your language. You must choose one of the available language feature codes and install that version instead. You must manually change the system library list to use IBM MQ in that language load.

  ```
  CHGSYSLIBL LIB(QSYS2924)
  ```
- If you are using Korean DBCS and you configure your terminal emulators to 24*80 sessions you might find that EDTF incorrectly displays DBCS characters in MQ error log messages that extend beyond 80 columns. To avoid this, configure your terminal emulators to use sessions capable of displaying 132 columns, for example 27*132.
- Issue the following command specifying the appropriate language ID:

  ```
  RSTLICPGM LICPGM(5724H72) DEV( installation device ) RSTOBJ(*LNG) LNG( language ID )
  ```

  This installs the commands, message file, and panel groups into the relevant QSYS library for the language. For example, library QSYS2928 is used for French. If this QSYS29nn library does not exist, it is created by the RSTLICPGM command.

5. To ensure that the product has loaded correctly, issue the Display Software Resources (DSPSFWRSC) command and check that the licensed program 5724H72 is listed. If you have installed the base and the optional samples, you see:

   ```
   Resource
   ID    Option Feature Description
   5724H72  *BASE  5050  IBM MQ for IBM i
   5724H72  *BASE  2924  IBM MQ for IBM i
   5724H72  1    5050  IBM MQ for IBM i - Samples
   ```

6. Press F11, while viewing the Display Software Resources screen, and you see the library and version number of the products installed:

   ```
   Resource          Feature
   ID    Option Feature  Type  Library  Release
   5724H72  *BASE  5050   *CODE  QMQM   V9R0M0
   5724H72  *BASE  2924   *LNG   QMQM   V9R0M0
   5724H72  1    5050   *CODE  QMQMSAMP V9R0M0
   ```

7. If you have installed additional language versions, you also see entries for these versions. For example, if you have installed the French version, for which the language ID is 2928, you see:

   a.

   ```
   Resource
   ID    Option Feature Description
   5724H72  *BASE  2928  IBM MQ for IBM i
   ```

   b. and when you press F11:

   ```
   Resource          Feature
   ID    Option Feature  Type  Library   Release
   5724H72  *BASE  2928  *LNG  QSYS2928  V9R0M0
   ```

8. Use the command DSPMQMVER to check exactly what version you have installed. For V9R0M0, it reports:

   ```
   Version: 9.0.0.0
   ```

9. Do the post installation tasks of checking for updates, checking program authorities and starting the IBM MQ subsystem, see "Performing post installation tasks for IBM MQ on IBM i" on page 314.

What to do next

If you want to see how the installation went in more detail, perform one or more of the following tasks:
- View the log file using the `DSPJOBLOG` command.
- View the spoolfile generated from the `RSTLICPGM` command.

If the installation of IBM MQ fails, see "Handling installation failures for IBM i" on page 315.

Related concepts:
"Uninstalling IBM MQ for IBM i" on page 325
There are two ways of uninstalling IBM MQ for IBM i.

Installing IBM MQ server silently on IBM i

IBM i

You can perform a non-interactive installation of IBM MQ using the `CALL PGM(QSYS/QLPACAGR)` command. A non-interactive installation is also known as a silent, or unattended installation.

Before you begin

Before you start the installation procedure, make sure that you have completed the necessary steps outlined in "Preparing the system on IBM i" on page 301.

About this task

This topic describes the non-interactive installation of a server.

Procedure

1. Pre-agree the license terms and conditions for the base by running the command,
   ```
   CALL PGM  ( QSYS/QLPACAGR) PARM ('5724H72' 'V8R0M0' '0000' 0)
   ```
   Where the parameters of **PARM** are,

   5724H72
   The product identifier for IBM i.

   V9R0M0
   The version, release, and modification level.

   0000
   The option number for the IBM MQ product.

   0 Unused error structure.
2. Optionally pre-agree the license terms and conditions for the samples by running the command,
   ```
   CALL PGM (QSYS/QLPACAGR) PARM ('5724H72' 'V8R0M0' '0001' 0)
   ```
   Where the parameters of **PARM** are,

   5724H72
   The product identifier for IBM i.

   V9R0M0
   The version, release, and modification level.

   0001
   The option number for the IBM MQ product.

   0 Unused error structure.
3. Install IBM MQ for IBM i, V9.0 base product, and primary language.

```
RSTLICPGM LICPGM (5724H72) DEV (installation device) OPTION (*BASE) OUTPUT (*PRINT)
```

where the parameters of RSTLICPGM are,

LICPGM(5724H72)
The product identifier for IBM i.

DEV(*installation device*)
The device from which the product is to be loaded, typically an optical drive, for example, `OPT01`.

OPTION (*BASE)
Install the base IBM MQ for IBM i product.

Unspecified parameters
Unspecified parameters, such as **RSTOBJ** (`*ALL`), revert to defaults. The command installs both IBM MQ and the language files for the primary language of your system. For installing additional languages, see step 4.

4. Optional: Install the samples using the command:

```
RSTLICPGM LICPGM (5724H72) DEV (installation device) OPTION (1) OUTPUT (*PRINT)
```

Where the parameters of RSTLICPGM are,

LICPGM (5724H72)
The product identifier for IBM i.

DEV (*installation device*)
The device from which the product is to be loaded, typically an optical drive, for example, `OPT01`.

OPTION (1)
Install the samples for IBM i.

OUTPUT (*PRINT)
The output is printed with the spooled output of the job.

5. Optional: To install additional languages, sign on to the system with a user profile that has *ALLOBJ special authority. Choose a language code from the table.

Table 43. Globalizations of IBM MQ for IBM i.

Language ID	Language
2909	Belgian English
2966	Belgian French MNCS (Multi-National Character Set)
2980	Brazilian Portuguese
2981	Canadian French MNCS
2975	Czech
2924	English uppercase and lowercase
2984	English US DBCS
2938	English US uppercase DBCS
2928	French
2940	French MNCS
2929	German
2939	German MNCS
2976	Hungarian
2932	Italian

Table 43. Globalizations of IBM MQ for IBM i. (continued)

Language ID	Language
2942	Italian MNCS
2962	Japanese
2930	Japanese Universal
2986	Korean
2978	Polish
2979	Russian
2989	Simplified Chinese
2931	Spanish

- If installing Japanese language feature code 2962, ensure the CCSID of the job installing the product is set to 939 and not 930. Do this to avoid problems with invariant lowercase characters in CCSID 930

  ```
  CHGJOB CCSID(939)
  ```
- If the language feature code is not in the table then the product has not been translated into your language. You must choose one of the available language feature codes and install that version instead. You must manually change the system library list to use IBM MQ in that language load.

  ```
  CHGSYSLIBL LIB(QSYS2924)
  ```
- If you are using Korean DBCS and you configure your terminal emulators to 24*80 sessions you might find that EDTF incorrectly displays DBCS characters in MQ error log messages that extend beyond 80 columns. To avoid this, configure your terminal emulators to use sessions capable of displaying 132 columns, for example 27*132.
- Issue the following command specifying the appropriate language ID:

  ```
  RSTLICPGM LICPGM(5724H72) DEV( installation device ) RSTOBJ(*LNG) LNG( language ID )
  ```

  This installs the commands, message file, and panel groups into the relevant QSYS library for the language. For example, library QSYS2928 is used for French. If this QSYS29nn library does not exist, it is created by the RSTLICPGM command.

6. To ensure that the product has loaded correctly, issue the Display Software Resources (DSPSFWRSC) command and check that the licensed program 5724H72 is listed. If you have installed the base and the optional samples, you see:

```
Resource
ID    Option Feature Description
5724H72  *BASE  5050  IBM MQ for IBM i
5724H72  *BASE  2924  IBM MQ for IBM i
5724H72  1    5050  IBM MQ for IBM i - Samples
```

7. Press F11, while viewing the Display Software Resources screen, and you see the library and version number of the products installed:

```
Resource          Feature
ID    Option Feature  Type  Library  Release
5724H72  *BASE  5050   *CODE  QMQM   V9R0M0
5724H72  *BASE  2924   *LNG   QMQM   V9R0M0
5724H72  1    5050   *CODE  QMQMSAMP V9R0M0
```

8. If you have installed additional language versions, you also see entries for these versions. For example, if you have installed the French version, for which the language ID is 2928, you see:

   a.

```
Resource
ID    Option Feature Description
5724H72  *BASE  2928  IBM MQ for IBM i
```

b. and when you press F11:

```
Resource          Feature
ID    Option Feature  Type  Library   Release
5724H72  *BASE  2928  *LNG  QSYS2928  V9R0M0
```

9. Use the command DSPMQMVER to check exactly what version you have installed. For V9R0M0, it reports:

```
Version: 9.0.0.0
```

10. Do the post installation tasks of checking for updates, checking program authorities and starting the IBM MQ subsystem, see "Performing post installation tasks for IBM MQ on IBM i" on page 314.

What to do next

If you want to see how the installation went in more detail, perform one or more of the following tasks:

- View the log file using the `DSPJOBLOG` command.
- View the spoolfile generated from the `RSTLICPGM` command.

If the installation of IBM MQ fails, see "Handling installation failures for IBM i" on page 315.

Installing Managed File Transfer on IBM i

IBM i

Install IBM MQ Managed File Transfer for IBM i by installing IBM MQ Java Messaging and Web Services server in its primary language, and installing additional options.

Before you begin

Note: Installing the latest version of IBM MQ Managed File Transfer includes client capabilities.

You have completed planning the installation, obtained the installation CDs and set the system values, see "Configuring and tuning the operating system on IBM i" on page 302.

You have installed the following components:

Table 44. Software requirements for IBM MQ Managed File Transfer

Program	Option	Description
5761JV1	14 or 15	Java SE 7 32 bit or Java SE 7 64 bit
5770SS1	39	International Components for Unicode
5724L26	*BASE	IBM MQ Java Messaging and Web Services

About this task

How to install base Managed File Transfer in its primary language, and install the other options.

You can install only one instance of Managed File Transfer for IBM i in each partition of your server.

Procedure

1. Sign on to the system with a user profile that has *ALLOBJ special authority, for example QSECOFR.
2. Install Managed File Transfer for IBM i, V9.0 base product.

   ```
   RSTLICPGM LICPGM (5725M50) DEV (installation device) OPTION (*BASE) OUTPUT (*PRINT)
   ```

   where the parameters of RSTLICPGM are,

   LICPGM (5725M50)
   The product identifier for Managed File Transfer for IBM i.

   DEV (*installation device*)
   The device from which the product is to be loaded, typically an optical drive, for example, OPT01.

   OPTION (*BASE)
   Install Managed File Transfer for IBM i for the IBM MQ base product.

   Unspecified parameters
   Unspecified parameters such as **RSTOBJ** (*ALL), revert to defaults. The command installs both IBM MQ and the language files for the primary language of your system.
3. Optional: Install the tools using the command:

   ```
   RSTLICPGM LICPGM(5725M50) DEV(installation device) OPTION(2) OUTPUT(*PRINT)
   ```

   Where the parameters of RSTLICPGM are,

   LICPGM (5725M50)
   The product identifier for Managed File Transfer for IBM i.

   DEV (*installation device*)
   The device from which the product is to be loaded, typically an optical drive, for example, OPT01.

   OPTION (2)
   Install the tools for Managed File Transfer for IBM i.

   OUTPUT (*PRINT)
   The output is printed with the spooled output of the job.

   Repeat step 3 for options 3 (agent) and 4 (services)
4. To ensure that the product has loaded correctly, issue the Display Software Resources (DSPSFWRSC) command and check that the licensed program 5725M50 is listed. If you have installed the base and the optional tools, you see:

   ```
   Resource
   ID       Option  Feature  Description
   5725M50  *BASE   5050     Managed File Transfer for IBM i
   5725M50  *BASE   2924     Managed File Transfer for IBM i
   5725M50  2       5050     Managed File Transfer for IBM i - Tools
   ```

5. Press F11, while viewing the Display Software Resources screen, and you see the library and version number of the products installed:

   ```
   Resource
   ID       Option  Feature  Type   Library  Release
   5725M50  *BASE   5050     *CODE  QMQMMFT  V9R0M0
   5725M50  *BASE   2924     *LNG   QMQMMFT  V9R0M0
   5725M50  2       5050     *CODE  MFTTOOL  V9R0M0
   ```

6. Do the post installation tasks of checking for updates, checking program authorities, and starting the Managed File Transfer subsystem.

What to do next

If you want to see how the installation went in more detail, perform one or more of the following tasks:
- View the log file using the `DSPJOBLOG` command.
- View the spoolfile generated from the `RSTLICPGM` command.

If the installation of IBM MQ fails, see "Handling installation failures for IBM i" on page 315.

Installing IBM MQ for IBM i from an Electronic Software Download

IBM i

You can perform an installation of IBM MQ for IBM i Version 9.0 from an installation image downloaded from IBM.

Before you begin

Before you start the installation procedure, make sure that you have completed the necessary steps outlined in "Preparing the system on IBM i" on page 301.

About this task

Two installation images are provided as zip files, a client and server image. These images contain all the licensed programs, and a client only image for the clients only.

The client and server image contains all seven compressed IBM i save files (**`SAVF`**), while the client image contains four save files. The save files are:
- MQ90BASE - IBM MQ client and server base program objects
- MQ90SAMP - IBM MQ client & server samples
- MQ90EN24 - IBM MQ client and server English US (2924) language objects

plus the client only images:
- MQ90CBASE - IBM MQ client
- MQ90CSAMP - IBM MQ client samples
- MQ90JBASE - IBM MQ Java
- MQ90JSAMP - IBM MQ Java samples

Procedure

1. Download one of the installation images and extract it to a temporary directory.
2. On IBM i, create a library containing sufficient empty save files to hold the uploaded files by using the commands:

   ```
   CRTLIB LIB(MQ90PROD)
   CRTSAVF FILE(MQ90PROD/MQ90BASE) /* Server and Client */
   CRTSAVF FILE(MQ90PROD/MQ90SAMP) /* Server and Client Samples */
   CRTSAVF FILE(MQ90PROD/MQ90EN24) /* 2924 English */
   CRTSAVF FILE(MQ90PROD/MQ90CBASE) /* Standalone Client */
   CRTSAVF FILE(MQ90PROD/MQ90CSAMP) /* Standalone Client Samples */
   CRTSAVF FILE(MQ90PROD/MQ90JBASE) /* Java and JMS Classes */
   CRTSAVF FILE(MQ90PROD/MQ90JSAMP) /* Java and JMS Samples */
   ```

   For additional languages

```
CRTSAVF FILE(MQ90PROD/MQ90EN09) /* 2909 Belgian English */
CRTSAVF FILE(MQ90PROD/MQ90FR28) /* 2928 French */
CRTSAVF FILE(MQ90PROD/MQ90JA30) /* 2930 Japanese */
CRTSAVF FILE(MQ90PROD/MQ90ES31) /* 2931 Spanish */
CRTSAVF FILE(MQ90PROD/MQ90IT32) /* 2932 Italian */
CRTSAVF FILE(MQ90PROD/MQ90EN38) /* 2938 English DBCS UPPERCASE */
CRTSAVF FILE(MQ90PROD/MQ90FR40) /* 2940 French MNCS */
CRTSAVF FILE(MQ90PROD/MQ90IT42) /* 2942 Italian MNCS */
CRTSAVF FILE(MQ90PROD/MQ90FR66) /* 2966 French MNCS */
CRTSAVF FILE(MQ90PROD/MQ90FR81) /* 2981 French MNCS */
CRTSAVF FILE(MQ90PROD/MQ90EN84) /* 2984 English DBCS */
CRTSAVF FILE(MQ90PROD/MQ90CZ75) /* 2975 Czech */
CRTSAVF FILE(MQ90PROD/MQ90HU76) /* 2976 Hungarian */
CRTSAVF FILE(MQ90PROD/MQ90PL78) /* 2978 Polish */
CRTSAVF FILE(MQ90PROD/MQ90RU79) /* 2979 Russian */
CRTSAVF FILE(MQ90PROD/MQ90PT80) /* 2980 Portugese/Brazilian */
CRTSAVF FILE(MQ90PROD/MQ90JA62) /* 2962 Japanese */
CRTSAVF FILE(MQ90PROD/MQ90KO86) /* 2986 Korean */
CRTSAVF FILE(MQ90PROD/MQ90ZH89) /* 2989 Chinese */
CRTSAVF FILE(MQ90PROD/MQ90DE29) /* 2929 German */
CRTSAVF FILE(MQ90PROD/MQ90DE39) /* 2939 German */
```

3. Start an ftp session to your IBM i machine and upload the required save files with the commands:

```
ftp (your_ibmi_hostname)
bin
put MQ90BASE MQ90PROD/MQ90BASE
put MQ90SAMP MQ90PROD/MQ90SAMP
put MQ90EN24 MQ90PROD/MQ90EN24
put MQ90CBASE MQ90PROD/MQ90CBASE
put MQ90CSAMP MQ90PROD/MQ90CSAMP
put MQ90JBASE MQ90PROD/MQ90JBASE
put MQ90JSAMP MQ90PROD/MQ90JSAMP
```

For additional language loads:

```
put MQ90EN09 MQ90PROD/MQ90EN09
put MQ90FR28 MQ90PROD/MQ90FR28
put MQ90JA30 MQ90PROD/MQ90JA30
put MQ90ES31 MQ90PROD/MQ90ES31
put MQ90IT32 MQ90PROD/MQ90IT32
put MQ90EN38 MQ90PROD/MQ90EN38
put MQ90FR40 MQ90PROD/MQ90FR40
put MQ90IT42 MQ90PROD/MQ90IT42
put MQ90FR66 MQ90PROD/MQ90FR66
put MQ90FR81 MQ90PROD/MQ90FR81
put MQ90EN84 MQ90PROD/MQ90EN84
put MQ90CZ75 MQ90PROD/MQ90CZ75
put MQ90HU76 MQ90PROD/MQ90HU76
put MQ90PL78 MQ90PROD/MQ90PL78
put MQ90RU79 MQ90PROD/MQ90RU79
put MQ90PT80 MQ90PROD/MQ90PT80
put MQ90JA62 MQ90PROD/MQ90JA62
put MQ90KO86 MQ90PROD/MQ90KO86
put MQ90ZH89 MQ90PROD/MQ90ZH89
put MQ90DE29 MQ90PROD/MQ90DE29
put MQ90DE39 MQ90PROD/MQ90DE39
```

4. To prepare for installation of IBM MQ for IBM i, sign on to your IBM i machine and ensure that you have followed the instructions detailed in "Preparing the system on IBM i" on page 301.
5. Enter the **RSTLICPGM** commands, specifying the installation device as *SAVF and naming the save file containing the options that you want to install. The IBM MQ Java licensed program can be installed stand-alone or can coexist with any of the other licensed programs.

   The IBM MQ client can be installed standalone, but it can only coexist with the IBM MQ Java on the same system.

Attempting to install the IBM MQ server on a system where the IBM MQ client is already installed performs a slip installation upgrade, replacing the client with the server licensed program.

Attempting to install the IBM MQ client stand-alone over the top of an existing server licensed program is not possible, and the installation fails.

For example:

```
/* IBM MQ Client and Server program objects */
RSTLICPGM LICPGM(5724H72) DEV(*SAVF) SAVF(MQ90PROD/MQ90BASE) +
RSTOBJ(*PGM) OPTION(*BASE) OUTPUT(*PRINT)

/* IBM MQ Client & Server English 2924 Language Load */
RSTLICPGM LICPGM(5724H72) DEV(*SAVF) SAVF(MQ90PROD/MQ90EN24) +
RSTOBJ(*LNG) LNG(2924) OUTPUT(*PRINT)

/* Additional languages - alter SAVF and LNG parameters... */
/* IBM MQ Client & Server Japanese 2930 Language Load */
RSTLICPGM LICPGM(5724H72) DEV(*SAVF) SAVF(MQ90PROD/MQ90JA30) +
RSTOBJ(*LNG) LNG(2930) OUTPUT(*PRINT)

/* IBM MQ Client & Server Samples */
RSTLICPGM LICPGM(5724H72) DEV(*SAVF) SAVF(MQ90PROD/MQ90SAMP) +
OPTION(1) OUTPUT(*PRINT)

/* IBM MQ Java */
RSTLICPGM LICPGM(5724L26) DEV(*SAVF) SAVF(MQ90PROD/MQ90JBASE) +
OPTION(*BASE) OUTPUT(*PRINT)

/* IBM MQ Java Samples */
RSTLICPGM LICPGM(5724L26) DEV(*SAVF) SAVF(MQ90PROD/MQ90JSAMP) +
OPTION(1) OUTPUT(*PRINT)

/* IBM MQ Client */
RSTLICPGM LICPGM(5725A49) DEV(*SAVF) SAVF(MQ90PROD/MQ90CBASE) +
OPTION(*BASE) OUTPUT(*PRINT)

/* IBM MQ Client Samples */
RSTLICPGM LICPGM(5725A49) DEV(*SAVF) SAVF(MQ90PROD/MQ90CSAMP) +
OPTION(1) OUTPUT(*PRINT)
```

6. Do the post installation tasks of checking for updates, checking program authorities and starting the IBM MQ subsystem, see "Performing post installation tasks for IBM MQ on IBM i" on page 314.

What to do next

If you want to see how the installation went in more detail, perform one or more of the following tasks:

- View the log file using the `DSPJOBLOG` command.
- View the spoolfile generated from the `RSTLICPGM` command.

If the installation of IBM MQ fails, see "Handling installation failures for IBM i" on page 315.

Performing post installation tasks for IBM MQ on IBM i

IBM i

Tasks to perform after you have installed IBM MQ for IBM i, and before using it.

About this task

When you have correctly installed IBM MQ for IBM i on your system:

Procedure

1. See the IBM MQ website at: http://www.ibm.com/software/products/ibm-mq for the latest product information.
2. Install and apply all fix packs.
3. Where you have more than one system and a mixture of releases of OS/400 or IBM i, and IBM MQ, you must take care when compiling CL programs. You must compile CL programs either on the system they are to run on, or on one with an identical combination of releases of OS/400 or IBM i, and IBM MQ. When you install later versions of IBM MQ, delete all IBM MQ commands from previous releases in any QSYSVvRrMm libraries using the QSYS/DLTCMD command.
4. If you have not installed IBM MQ on your system before, you must add user profiles to the `QMQMADM` group profile. Make all user profiles that are to be used for creating and administering queue managers members of the `QMQMADM` group profile, using the command `CHGUSRPRF`.
   a. Start the IBM MQ subsystem, by issuing the command:
      ```
      STRSBS SBSD(QMQM/QMQM)
      ```
      Note: The subsystem must be started after each IPL of the system, so you might choose to start it as part of your system startup process.
5. Create the system-default objects. The system-default objects are created automatically when you issue the CRTMQM command to create a queue manager. For example: `CRTMQM MQMNAME(QMGRNAME) ASP(*SYSTEM)`. You can refresh them using the STRMQM command (Warning: this command will replace any existing default objects). For example: `STRMQM MQMNAME(QMGRNAME) RDEFSYS(*YES)`. Refer to the onscreen help for information about using this command.

   Note: on the command `STRMQM MQMNAME(QMGRNAME) RDEFSYS(*YES)`:
   - The command does not re-create the objects, it performs a CRTxxxx REPLACE(*YES) for all of the SYSTEM.* objects.
   - This means that it refreshes the parameters on the objects back to their defaults. So if, for example, on the SYSTEM.DEFAULT.LOCAL.QUEUE object, TRGENBL had previously been changed to *YES, then, when the command is run, it is changed back to TRGENBL(*NO).
   - If any messages exist on a queue, they are not removed, because the queues are not physically deleted.
   - The contents of the SYSTEM.AUTH.DATA.QUEUE are untouched when this command is run.
   - So, if the contents of this (or any other significant queue) become corrupt, it must be physically deleted and re-created either from scratch, or from a backup.

Results

You are now ready to start using IBM MQ for IBM i.

Note: When you install IBM MQ for IBM i, two user profiles are created:
- QMQM
- QMQMADM

These two objects are central to the correct running of IBM MQ for IBM i. Do not alter or delete them. If you do, IBM cannot guarantee correct behavior of your product.

If you uninstall IBM MQ and data, these profiles are deleted. If you uninstall IBM MQ only, these profiles are retained.

Handling installation failures for IBM i

IBM i

If the installation of IBM MQ Server or Client for IBM i fails, you must remove the installed and partially installed objects before attempting reinstallation.

Procedure

1. Delete installed options using `DLTLICPGM LICPGM(5725A49)OPTION(*ALL)`.
2. Delete partially installed options by deleting the `QMQM` library (and the `QMQMSAMP` libraries if necessary).
3. Delete the IFS directory `/QIBM/ProdData/mqm` and its subdirectories using the `EDTF` command, for example: `EDTF STMF('/QIBM/ProdData')` and select **option 9** for the `mqm` directory.

   If the installation of IBM MQ Java fails, remove the partly installed objects before attempting reinstallation:

   a. Delete the `QMQMJAVA` library.
   b. Delete the IFS directory `/QIBM/ProdData/mqm/java` and its subdirectories using the **`EDTF`** command, for example:

      ```
      EDTF STMF ('/QIBM/ProdData/mqm')
      ```

      Select option 9 against the Java directory.

Converting a trial license on IBM i

IBM i

Convert a trial license to a full license without reinstalling IBM MQ.

When the trial license expires, the “count-down” displayed by the **`strmqm`** command informs you the license has expired, and the command does not run.

Before you begin

1. IBM MQ is installed with a trial license.
2. You have access to the installation media of a fully licensed copy of IBM MQ.

About this task

Run the **`setmqprd`** command to convert a trial license to a full license.

If you do not want to apply a full license to your trial copy of IBM MQ, you can uninstall it at any time.

Procedure

1. Obtain the full license from the fully licensed installation media.

   The full license file is `amqpcert.lic`. On IBM i, issue the command

   ```
   CALL PGM(QMQM/SETMQPRD) PARM('/QOPT/OPT01/amqpcert.lic')
   ```
2. Run the **`setmqprd`** command from the installation that you are upgrading:

   ```
   MQ_INSTALLATION_PATH/bin/setmqprd /MediaRoot/licenses/amqpcert.lic
   ```

Related information:
setmqprd

Installing an IBM MQ client on IBM i

IBM i

The IBM MQ client for IBM i is a part of the IBM MQ product.

Before you begin

Attention: If you have already installed the IBM MQ server, you already have a client and must not attempt to install the stand-alone client.

You can install only one instance of IBM MQ client for IBM i in each partition of your server.

When you install IBM MQ client for IBM i two user profiles are created:
- QMQM
- QMQMADM

These two objects are central to the correct running of IBM MQ for IBM i. Do not alter or delete them. If you do, IBM cannot guarantee correct behavior of your product. These profiles are retained when the product is deleted.

About this task

This procedure covers the installation of both the client and the client samples. If you do not want to install the client samples, then do not complete the steps specific to the samples.

After following the optional step to pre-agree the license, and then issuing the **RSTLICPGM** command, the installation runs without requiring any interactive input.

Procedure

1. Sign on to the system with a user profile that has `*ALLOBJ` special authority, for example `QSECOFR`.
2. Optional: Pre-agree the license terms and conditions. If you do not choose to pre-agree the license, the license agreement is displayed for you to accept. Run the following commands to pre-agree the license terms and conditions:
   a. For the client:
   ```
   CALL PGM (QSYS/QLPACAGR) PARM ('5725A49' 'V8R0M0' '0000' 0)
   ```
   The parameters of **PARM** are:

   5725A49
   The product identifier for IBM MQ client for IBM i

   V8R0M0
   The version, release, and modification level

   0000
   The option number for the base IBM MQ client for IBM i product

   0 Unused error structure

   b. For the client samples:
   ```
   CALL PGM (QSYS/QLPACAGR) PARM ('5725A49' 'V8R0M0' '0001' 0)
   ```
   The parameters of **PARM** are:

5725A49
The product identifier for IBM MQ client for IBM i

V8R0M0
The version, release, and modification level

0001
The option number for the samples

0 Unused error structure

3. Issue the installation command to run the installation without requiring any interactive input:
   a. Install the client by issuing the following command:
      ```
      RSTLICPGM LICPGM (5725A49) DEV (installation device) OPTION (*BASE) OUTPUT (*PRINT)
      ```
      The parameters of RSTLICPGM are:

      LICPGM (5725A49)
      The product identifier for IBM MQ client for IBM i

      DEV (*installation device*)
      The device from which the product is to be loaded, typically an optical drive, for example, `OPT01`

      OPTION (*BASE)
      The level of IBM MQ client for IBM i product installed

      OUTPUT (*PRINT)
      Whether the spooled output of the job is printed
   b. Install the samples by issuing the following command:
      ```
      RSTLICPGM LICPGM (5725A49) DEV (installation device) OPTION (1) OUTPUT (*PRINT)
      ```
      The parameters of RSTLICPGM are:

      LICPGM (5725A49)
      The product identifier for IBM MQ client for IBM i

      DEV (*installation device*)
      The device from which the product is to be loaded, typically an optical drive, for example, `OPT01`

      OPTION (1)
      The samples option

      OUTPUT (*PRINT)
      Whether the spooled output of the job is printed
4. To ensure that the product has loaded correctly, issue the Display Software Resources (**DSPSFWRSC**) command and check that the licensed program 5725A49 is listed. If you have installed the base and the optional samples, you see:
   ```
   Resource
   ID    Option Feature Description
   5725A49  *BASE  5050   IBM MQ client for IBM i
   5725A49  1    5050   IBM MQ client for IBM i -Samples
   ```
5. To see the library and version number of the products installed, press **F11**, while viewing the Display Software Resources screen. The following screen is displayed:
   ```
   Resource          Feature
   ID    Option Feature  Type  Library  Release
   5725A49  *BASE  5050   *CODE  QMQM    V8R0M0
   5725A49  1    5050   *CODE  QMQMSAMP  V8R0M0
   ```

6. To check exactly what version you have installed, use the **DSPMQMVER** program. For example, `CALL PGM(QMQM/DSPMQVER)` from the command line or `/QSYS.LIB/QMQM.LIB/DSPMQVER.PGM -a` in a qshell.

What to do next

If you want to see how the installation went in more detail, perform one or more of the following tasks:
- View the log file using the `DSPJOBLOG` command.
- View the spoolfile generated from the `RSTLICPGM` command.

If the installation of IBM MQ client for IBM i failed, see "Handling installation failures for IBM i" on page 315

Related concepts:

"Uninstalling IBM MQ for IBM i" on page 325
There are two ways of uninstalling IBM MQ for IBM i.

Installation of IBM MQ client and IBM MQ server for IBM i

When you install an IBM MQ server on an IBM i system, the client is also automatically installed.

The installed version of the IBM MQ client for IBM i can be refreshed by using a "slip installation" which replaces an existing installation with a fresh image.

Installing a client over an existing client results in a successful installation.

Installing a client over an existing server results in a failure with a CPDB6A4 error.

Installing a server over an existing client results in a successful upgrade of the client to both server and client capabilities.

Installing IBM MQ Java messaging and web services for IBM i

Install IBM MQ Java messaging and web services for IBM i from either product CD, using the **RSTLICPGM** command.

Before you begin

You can install only one instance of IBM MQ Client for IBM i in each partition of your server.

If you have Java messaging and web services v7.0 or v7.1 installed and want to install v8.0, you can install the new version without uninstalling the old one.

If you have MA88 installed, and try to install anyway, the installation fails with a warning requesting you to uninstall the old client. To uninstall MA88, issue the following command:

```
DLTLICPGM LICPGM(5648C60) OPTION(*ALL)
```

If this command fails to delete the IFS directory `/QIBM/ProdData/mqm/java` and its subdirectories, use the EDTF command and select option 9 against the Java directory. For example:

```
EDTF STMF('/QIBM/ProdData/mqm')
```

About this task

This procedure covers the installation of both the Java messaging and web services, and the Java messaging and web services samples. If you do not want to install the samples, then do not complete the steps specific to the samples.

After following the optional step to pre-agree the license, and then issuing the **RSTLICPGM** command, the installation runs without requiring any interactive input.

Procedure

1. Sign on to the system with a user profile that has `*ALLOBJ` special authority, for example `QSECOFR`.
2. Optional: Pre-agree the license terms and conditions. If you do not choose to pre-agree the license, the license agreement is displayed for you to accept. Run the following commands to pre-agree the license terms and conditions:
   a. For Java messaging and web services:
      ```
      CALL PGM (QSYS/QLPACAGR) PARM ('5724L26' 'V8R0M0' '0000' 0)
      ```

      The parameters of **PARM** are:

      5724L26
      The product identifier for IBM MQ Java messaging and web services for IBM i

      V8R0M0
      The version, release, and modification level

      0000
      The option number for the base IBM MQ Java messaging and web services product.

      0 Unused error structure
   b. For the samples:
      ```
      CALL PGM (QSYS/QLPACAGR) PARM ('5724L26' 'V8R0M0' '0001' 0)
      ```

      The parameters of **PARM** are:

      5724L26
      The product identifier for IBM MQ Java messaging and web services for IBM i

      V8R0M0
      The version, release, and modification level

      0001
      The option number for the samples.

      0 Unused error structure
3. Issue the installation command to run the installation without requiring any interactive input:
   a. Install the IBM MQ Java messaging and web services by issuing the following command:
      ```
      RSTLICPGM LICPGM (5724L26) DEV (installation device) OPTION (*BASE) OUTPUT (*PRINT)
      ```

      The parameters of RSTLICPGM are:

      LICPGM (5724L26)
      The product identifier for IBM MQ Java messaging and web services for IBM i

      DEV (*installation device*)
      The device from which the product is to be loaded, typically an optical drive, for example, `OPT01`

      OPTION (*BASE)
      Install the base IBM MQ Java messaging and web services for IBM i

OUTPUT (*PRINT)
Whether the spooled output of the job is printed

b. Install the samples by issuing the following command:

```
RSTLICPGM LICPGM (5724L26) DEV (installation device) OPTION (1) OUTPUT (*PRINT)
```

The parameters of RSTLICPGM are:

LICPGM (5724L26)
The product identifier for IBM MQ Java messaging and web services for IBM i

DEV (*installation device*)
The device from which the product is to be loaded, typically an optical drive, for example, `OPT01`

OPTION (1)
Install the samples

OUTPUT (*PRINT)
Whether the spooled output of the job is printed

4. To ensure that the product has loaded correctly, issue the Display Software Resources (DSPSFWRSC) command and check that the licensed program `5724L26` is listed. If you have installed the base and the optional samples, you see:

```
Resource
ID    Option Feature Description
5724L26  *BASE  5050  IBM MQ Java Messaging and Web Services
5724L26  1    5050  IBM MQ Java Messaging and Web Services - Samp
```

5. Press **F11** while viewing the Display Software Resources screen, and you see the library and version number of the products installed:

```
Resource          Feature
ID    Option Feature  Type  Library  Release
5724L26  *BASE  5050   *CODE  QMQMJAVA V8R0V0
5724L26  1    5050   *CODE  QMQMJAVA V8R0V0
```

6. Check what versions you have installed by using the following commands:
IBM MQ Classes for Java:

```
java com.ibm.mq.MQJavaLevel
```

Note: For this command to work, you might have to set your environment classpath to:
• `/QIBM/ProdData/mqm/java/lib/com.ibm.mq.jar`

IBM MQ Classes for Java Message Service:

```
java com.ibm.mq.jms.MQJMSLevel
```

Note: For this command to work, you might need to set your environment classpath to:
• `/QIBM/ProdData/mqm/java/lib/com.ibm.mqjms.jar`

See Environment variables relevant to IBM MQ classes for Java and Environment variables used by IBM MQ classes for JMS.

For v8.0, both report:

```
Version: 8.0.0.0
```

Note: The command uses the Java classes, and so it reports the version and also performs some verification that the classes are installed and working.

7. See the following topics for full details of verification of both:

- Using IBM MQ classes for Java
- Using IBM MQ classes for JMS

Verifying an IBM MQ installation on IBM i

IBM i

The topics in this section provide instructions on how to verify a client installation of IBM MQ on IBM i systems.

Verifying a client installation using the command line on IBM i

IBM i

You can verify a client installation using the command line. On the server you create a queue manager, a local queue, a listener, and a server-connection channel. You must also apply security rules to allow the client to connect and make use of the queue defined. On the client you create a client-connection channel, and then use the sample PUT and GET programs to complete the verification procedure.

The verification procedure shows how to create a queue manager called `queue.manager.1`, a local queue called `QUEUE1`, and a server-connection channel called `CHANNEL1` on the server.

It shows how to create the client-connection channel on the IBM MQ MQI client workstation. It then shows how to use the sample programs to put a message onto a queue, and get the message from the queue.

The example does not address any client security issues. See Setting up IBM MQ MQI client security for details if you are concerned with IBM MQ MQI client security issues.

The verification procedure assumes that:
- The full IBM MQ server product has been installed on a server.
- The server installation is accessible on your network.
- The IBM MQ MQI client software has been installed on a client system.
- The IBM MQ sample programs have been installed.
- TCP/IP has been configured on the server and client systems. For more information, see Configuring connections between the server and client.

First, set up the server using the command line, using the instructions in "Setting up the server using the command line IBM i" on page 322.

Once you have set up the server, you must set up the client, using the instructions in "Connecting to a queue manager, using the `MQSERVER` environment variable on IBM i" on page 323.

Finally, you can test the communications between client and server, using the instructions in "Testing communication between a client and a server on IBM i" on page 324.

Setting up the server using the command line IBM i: IBM i

Follow these instructions to create a queue manager, queue, and channel on the server. You can then use these objects to verify the installation.

About this task

These instructions assume that no queue manager or other IBM MQ objects have been defined.

IBM MQ object definitions are case-sensitive. Any text entered as an MQSC command in lowercase is converted automatically to uppercase unless you enclose it in single quotation marks. Make sure that you type the examples exactly as shown.

Procedure

1. Create a user ID on the server that is not in the `mqm` group. This user ID must exist on the server and client. This is the user ID that the sample applications must be run as, otherwise a 2035 error is returned.
2. Log in as a user in the MQM group.
3. Create a queue manager called `QUEUE.MANAGER.1` by entering the following command:
   ```
   crtmqm QUEUE.MANAGER.1
   ```
   You see messages telling you that the queue manager has been created.
4. Start the queue manager by entering the following command:
   ```
   strmqm QUEUE.MANAGER.1
   ```
   A message tells you when the queue manager has started.
5. Define a local queue called `QUEUE1` by entering the following command:
   ```
   CRTMQMQ QNAME(QUEUE1) QTYPE(*LCL)
   ```
   A message tells you when the queue has been created.
6. Allow the user ID that you created in step 1 to use `QUEUE1` by entering the following command:
   ```
   SET AUTHREC PROFILE(QUEUE1) OBJTYPE(QUEUE) PRINCIPAL(' non_mqm_user ') AUTHADD(PUT,GET)
   ```
   where *non_mqm_user* is the user ID created in step 1. A message tells you when the authorization has been set. You must also run the following command to give the user ID authority to connect:
   ```
   SET AUTHREC OBJTYPE(QMGR) PRINCIPAL(' non_mqm_user ') AUTHADD(CONNECT)
   ```
   If this command is not run, a 2305 stop error is returned.
7. Define a server-connection channel by entering the following command:
   ```
   CRTMQMCHL CHLNAME(CHANNEL1) CHLTYPE(*SVRCN) TRPTYPE(*TCP)
   MCAUSRID('QMQM')
   ```
   A message tells you when the channel has been created.
8. Allow your client channel to connect to the queue manager and run under the user ID that you created in step 1, by entering the following MQSC command:
   ```
   SET CHLAUTH(CHANNEL1) TYPE(ADDRESSMAP) ADDRESS(' client_ipaddr ') MCAUSER(' non_mqm_user ')
   ```
   where *client_ipaddr* is the IP address of the client system, and *non_mqm_user* is the user ID created in step 1. A message tells you when the rule has been set.
9. Define a listener by entering the following command:
   ```
   DEFINE LISTENER (LISTENER1) TRPTYPE (TCP) CONTROL (QMGR) PORT (port_number)
   ```

where *port_number* is the number of the port the listener is to run on. This number must be the same as the number used when defining your client-connection channel in "Installing an IBM MQ client on IBM i" on page 316.

Note: If you omit the port parameter from the command, a default value of `1414` is used for the listener port. If you want to specify a port other than `1414`, you must include the port parameter in the command, as shown.

10. Start the listener by entering the following command:
    ```
    STRMQMLSR MQMNAME('QUEUE.MANAGER.1') PORT(1414)
    ```
11. Stop MQSC by entering:
    ```
    end
    ```

    You see some messages, followed by the command prompt.

What to do next

Follow the instructions to set up the client. See "Connecting to a queue manager, using the `MQSERVER` environment variable on IBM i."

Connecting to a queue manager, using the `MQSERVER` environment variable on IBM i: IBM i

When an IBM MQ application is run on the IBM MQ MQI client, it requires the name of the MQI channel, the communication type, and the address of the server to be used. Provide these parameters by defining the `MQSERVER` environment variable.

Before you begin

Before you start this task, you must complete the task, "Setting up the server using the command line IBM i" on page 322, and save the following information:
- The host name or IP address of the server and port number that you specified when creating the listener.
- The channel name of the server-connection channel.

About this task

This task describes how to connect an IBM MQ MQI client, by defining the `MQSERVER` environment variable on the client.

Procedure

1. Log in as the userid that you created in Step 1 of "Setting up the server using the command line IBM i" on page 322.
2. Check the TCP/IP connection. From the client, enter one of the following commands:
   - `ping server-hostname`
   - `ping n.n.n.n`

     `n.n.n.n` represents the network address. You can set the network address in IPv4 dotted decimal form, for example, `192.0.2.0`. Alternatively, set the address in IPv6 hexadecimal form, for example `2001:0DB8:0204:acff:fe97:2c34:fde0:3485`.

   If the **`ping`** command fails, correct your TCP/IP configuration.
3. Set the `MQSERVER` environment variable. From the client, enter one the following command:
   ```
   ADDENVVAR ENVVAR(MQSERVER) VALUE('CHANNEL1/TCP/server-address (port)')
   ```

   Where:

- *CHANNEL1* is the server-connection channel name.
- *server-address* is the TCP/IP host name of the server.
- *port* is the TCP/IP port number the server is listening on.

If you do not give a port number, IBM MQ uses the one specified in the `qm.ini` file, or the client configuration file. If no value is specified in these files, IBM MQ uses the port number identified in the TCP/IP services file for the service name `MQSeries`. If an `MQSeries` entry in the services file does not exist, a default value of `1414` is used. It is important that the port number used by the client and the port number used by the server listener program are the same.

What to do next

Use the sample programs to test communication between the client and server; see "Testing communication between a client and a server on IBM i."

Testing communication between a client and a server on IBM i: IBM i

On the IBM MQ MQI client workstation, use the amqsputc sample program to put a message on the queue at the server workstation. Use the amqsgetc sample program to get the message from the queue back to the client.

Before you begin

Complete the previous topics in this section:
- Set up a queue manager, channels, and queue.
- Open a command window.
- Set system environment variables.

About this task

Note that IBM MQ object definitions are case-sensitive. Text entered as an MQSC command in lowercase is converted automatically to uppercase unless you enclose it in single quotation marks. Make sure that you type the examples exactly as shown.

Procedure

1. Start the PUT program for `QUEUE1` on `QUEUE.MANAGER.1` by entering the following command:
   ```
   CALL PGM(QMQM/AMQSPUTC) PARM(QUEUE1 QUEUE.MANAGER.1)
   ```
   If the command is successful, the following messages are displayed:
   ```
   Sample AMQSPUT0 start target queue is QUEUE1
   ```
   Tip: You might get the error, `MQRC_NOT_AUTHORIZED` (2035). By default, channel authentication is enabled when a queue manager is created. Channel authentication prevents privileged users accessing a queue manager as an IBM MQ MQI client. For verifying the installation, you can either change the MCA user ID to a non-privileged user, or disable channel authentication. To disable channel authentication run the following MQSC command:
   ```
   ALTER QMGR CHLAUTH(DISABLED)
   ```
   When you finish the test, if you do not delete the queue manager, re-enable channel authentication:
   ```
   ALTER QMGR CHLAUTH(ENABLED)
   ```
2. Type some message text, then press **Enter** twice. The following message is displayed:
   ```
   Sample AMQSPUT0 end
   ```

Your message is now on the queue that is on the server queue manager.

3. Start the GET program for `QUEUE1` on `QUEUE.MANAGER.1` by entering the following command:

   ```
   CALL PGM(QMQM/AMQSGETC) PARM(QUEUE1 QUEUE.MANAGER.1)
   ```

   The sample program starts, and your message is displayed. After a short pause (approximately 30 seconds), the sample ends and the command prompt is displayed again.

Results

You have now successfully verified the client installation.

What to do next

1. On the server, stop the queue manager by entering the following command:

   ```
   ENDMQM MQMNAME(QUEUE.MANAGER.1)
   ```

2. On the server, delete the queue manager by entering the following command:

   ```
   DLTMQM MQMNAME(QUEUE.MANAGER.1)
   ```

Uninstalling IBM MQ for IBM i

IBM i

There are two ways of uninstalling IBM MQ for IBM i.

To uninstall IBM MQ for IBM i, perform one of the following tasks:

- A *standard* deletion removes IBM MQ product code but preserves user data.
- An *entire* deletion removes both IBM MQ product code and user data.

Both types of deletion require you to be signed on to the system with a user profile that has *ALLOBJ special authority, for example, QSECOFR. Security administrator (*SECADM) special authority is also required to delete the QMQM and QMQMADM user profiles.

Related concepts:

"Reinstalling IBM MQ for IBM i" on page 329
You can reinstall IBM MQ for IBM i without losing any of your data.

Related tasks:

"Uninstalling IBM MQ but retaining data on IBM i"
If you want to retain your user data, for example, because you intend to reinstall IBM MQ for IBM i at a later date, you must perform a standard deletion of the product.

"Uninstalling IBM MQ and data on IBM i" on page 327
You can delete IBM MQ entirely, including all user data.

"Uninstalling IBM MQ Java Messaging and Web Services on IBM i" on page 328
Follow these instructions to uninstall IBM MQ Java.

"Uninstalling IBM MQ MQI client for IBM i" on page 328
If the IBM MQ MQI client for IBM i must be uninstalled, follow the correct procedure to ensure that all the relevant directories and files are removed.

Uninstalling IBM MQ but retaining data on IBM i

IBM i

If you want to retain your user data, for example, because you intend to reinstall IBM MQ for IBM i at a later date, you must perform a standard deletion of the product.

About this task

To perform a standard deletion of IBM MQ for IBM i, so that your user data is retained, complete the following steps:

Procedure

1. Quiesce IBM MQ for IBM i. For more information, see Quiescing IBM MQ for IBM i.
2. End the IBM MQ subsystem, by issuing the command:
   ```
   ENDSBS SBS(QMQM)
   ```
3. Ensure that no locks are held on the library QMQM, by issuing the command:
   ```
   WRKOBJLCK OBJ(QMQM) OBJTYPE(*LIB)
   ```
4. Use the Delete Licensed Program (DLTLICPGM) command to delete the base product (and also the samples if you chose to install them).

   To delete only the samples, issue the command:
   ```
   DLTLICPGM LICPGM( 5724H72 ) OPTION(1)
   ```
   To delete only extra language versions installed, issue the command:
   ```
   DLTLICPGM LICPGM(5724H72) LNG(nnnn)
   ```

   where *nnnn* is the language number, as in the list here:

Table 45. Globalizations of IBM MQ for IBM i.

Language ID	Language
2909	Belgian English
2966	Belgian French MNCS (Multi-National Character Set)
2981	Canadian French MNCS
2975	Czech
2950	English uppercase
2924	English uppercase and lowercase
2984	English US DBCS
2938	English US uppercase DBCS
2928	French
2940	French MNCS
2929	German
2939	German MNCS
2976	Hungarian
2932	Italian
2942	Italian MNCS
2962	Japanese
2986	Korean
2978	Polish
2979	Russian
2989	Simplified Chinese
2931	Spanish

   To delete the base product and the samples, issue the command:
   ```
   DLTLICPGM LICPGM( 5724H72 ) OPTION(*ALL)
   ```

Results

Deleting IBM MQ for IBM i in this way deletes only the objects that belong to IBM MQ: the QMQM library, the QMQM samp library, and the subdirectories that belong to IBM MQ server within the `/QIBM/ProdData/mqm` directory.

If that leaves no other subdirectories (for example, if IBM MQ Java is installed it uses subdirectories there) then the `/QIBM/ProdData/mqm` directory itself is deleted.

None of the queue manager journal libraries, or IFS directories based upon `/QIBM/UserData` are removed.

Uninstalling IBM MQ and data on IBM i

IBM i

You can delete IBM MQ entirely, including all user data.

About this task

Important: If you are going to delete IBM MQ entirely, including all user data, save your user data first. It cannot be recovered.

To delete IBM MQ for IBM i entirely, complete the following steps:

Procedure

1. Quiesce IBM MQ for IBM i. For more information, see Quiescing IBM MQ for IBM i.
2. Delete each queue manager in turn by using the command `WRKMQM` and selecting option `4`.
3. End the IBM MQ subsystem, by issuing the command:
   ```
   ENDSBS SBS(QMQM)
   ```
4. Ensure that no locks are held on the library QMQM, by issuing the command:
   ```
   WRKOBJLCK OBJ(QMQM) OBJTYPE(*LIB)
   ```
5. Optional: If you want to also uninstall IBM MQ Java, you can do it now, using the command:
   ```
   DLTLICPGM LICPGM( 5724L26 ) OPTION(*ALL)
   ```
   This will also uninstall the Java Samples, if they were installed.
6. Use the Delete Licensed Program (DLTLICPGM) command to delete the base product (and also the samples if you chose to install them). To delete the base product and the samples issue the command:
   ```
   DLTLICPGM LICPGM( 5724H72 ) OPTION(*ALL)
   ```
7. Delete the directory /QIBM/UserData/mqm and its subdirectories. Do this using the EDTF command and selecting option 9 (recursive delete) for the mqm directory, as follows,

   Note: If you do this, you no longer have any information regarding your installation. Use this command with extreme caution.

   The format of the command is:
   ```
   EDTF STMF('/QIBM/UserData')
   ```
   Alternatively, you can delete the /QIBM/UserData/mqm directory and its subdirectories by repeated use of the RMVLNK and RMVDIR commands.
8. Identify all the users who belong to the QMQMADM group. Use the DSPUSRPRF command to display a list of them. You must remove the QMQMADM group profile from their user profiles before you can delete the QMQMADM user profile. The format of the command is:
   ```
   DSPUSRPRF USRPRF(QMQMADM) TYPE(*GRPMBR)
   ```

9. You must alter the ownership or delete the objects. For each of the user profiles QMQM and QMQMADM, use the WRKOBJOWN command to list all the objects owned by the profile. The format of the command is:

   ```
   WRKOBJOWN USRPRF( PROFILE )
   ```

10. Delete the two user profiles. The format of the command is:

    ```
    DLTUSRPRF USRPRF(QMQM) OWNOBJOPT(*DLT)
    DLTUSRPRF USRPRF(QMQMADM) OWNOBJOPT(*DLT)
    ```

Uninstalling IBM MQ Java Messaging and Web Services on IBM i

IBM i

Follow these instructions to uninstall IBM MQ Java.

About this task

To uninstall the IBM MQ Java product.

Procedure

1. Make sure you are signed on to the system with a user profile that has *ALLOBJ special authority, for example QSECOFR.
2. Issue the command:

   ```
   DLTLICPGM LICPGM(5724L26) OPTION(*ALL)
   ```

Results

Deleting IBM MQ Java for IBM i deletes the objects that belong to it: the QMQMJAVA library, and the subdirectories that belong to IBM MQ Java within the `/QIBM/ProdData/mqm` directory.

If that leaves no other subdirectories (for example if the IBM MQ Server is installed it uses subdirectories there) then the `/QIBM/ProdData/mqm` directory itself is deleted.

Uninstalling IBM MQ MQI client for IBM i

IBM i

If the IBM MQ MQI client for IBM i must be uninstalled, follow the correct procedure to ensure that all the relevant directories and files are removed.

Procedure

1. Make sure you are signed on to the system with a user profile that has *ALLOBJ special authority, for example QSECOFR.
2. Use the Delete Licensed Program (**DLTLICPGM**) command to delete the IBM MQ MQI client for IBM i product (and also the samples if you chose to install them):

   To delete only the samples, issue the command

   ```
   DLTLICPGM LICPGM(5725A49) OPTION(1)
   ```

   To delete IBM MQ MQI client and the samples, issue the command:

   ```
   DLTLICPGM LICPGM(5725A49) OPTION(*ALL)
   ```

Results

Deleting IBM MQ MQI client for IBM i deletes the objects that belong to it - the QMQM library, and the subdirectories that belong to IBM MQ MQI client for IBM i within the `/QIBM/ProdData/mqm` directory. If that leaves no other subdirectories (for example if the IBM MQ Java Client for IBM i is installed it uses subdirectories there) then the `/QIBM/ProdData/mqm` directory itself is deleted.

Uninstalling Managed File Transfer on IBM i

Follow these instructions to uninstall Managed File Transfer on IBM i.

Before you begin

To uninstall IBM MQ Managed File Transfer for IBM i, perform one of the following tasks:

- A *standard* deletion removes Managed File Transfer product code but preserves user data.
- An *entire* deletion removes both Managed File Transfer product code and user data.

  Note that an entire deletion requires that you manually remove the configuration data in the `/QIBM/UserData/mqm/mqft` directory.

Both types of deletion require you to be signed on to the system with a user profile that has *ALLOBJ special authority, for example, QSECOFR.

About this task

To uninstall the Managed File Transfer product.

Procedure

1. Make sure you are signed on to the system with a user profile that has *ALLOBJ special authority, for example QSECOFR.
2. Issue the command:

   ```
   DLTLICPGM LICPGM(5725M50) OPTION(*ALL)
   ```

Results

Deleting Managed File Transfer for IBM i deletes the objects that belong to it: the QMQMMFT library, and the subdirectories that belong to Managed File Transfer within the `/QIBM/ProdData/mqm` directory.

Note that licence files are copied to `/QIBM/ProdData/mqm/properties/version`, and an uninstallation will delete files in this directory. However, files are left in `/QIBM/ProdData/mqm/properties/5725M50` as trash. For a clean uninstallation, you must delete the files in this directory.

Reinstalling IBM MQ for IBM i

You can reinstall IBM MQ for IBM i without losing any of your data.

When you reinstall IBM MQ for IBM i, the system checks whether the IBM MQ configuration file (mqs.ini) exists. If the file exists, it is kept and used with the newly installed system. If the file does not exist, an empty mqs.ini file is placed in the directory `/QIBM/UserData/mqm`.

All data that you have in the UserData directory is referenced by the newly installed system. In addition, all the queue manager-associated libraries containing journal and receiver information are referenced by the new system.

Related tasks:
"Installing IBM MQ server on IBM i" on page 303
Install IBM MQ for IBM i by installing the IBM MQ server in its primary language, installing samples and installing additional languages.

Installing and uninstalling IBM MQ on Linux

Linux

Installation tasks that are associated with installing IBM MQ on Linux are grouped in this section.

About this task

To prepare for installation and to install IBM MQ, complete the following tasks.

If product fixes or updates are made available, see IBM MQ maintenance tasks for information about how to apply these changes.

Procedure

- To install IBM MQ on Linux using rpm, see "Installing IBM MQ on Linux using rpm" on page 340.
- V 9.0.2 To install IBM MQ on Linux Ubuntu using a Debian installer , see "Installing IBM MQ on Linux Ubuntu using Debian" on page 363.

Checking requirements on Linux

Linux

Before you install IBM MQ on Linux, you must check for the latest information and system requirements.

About this task

A summary of the tasks that you must complete to check system requirements are listed here with links to further information.

Procedure

1. Check that you have the latest information, including information on hardware and software requirements. See "Where to find product requirements and support information" on page 195.
2. Check that your systems meet the initial hardware and software requirements for Linux. See "Hardware and software requirements on Linux systems" on page 331.

   The supported hardware and software environments are occasionally updated. See System Requirements for IBM MQ for the latest information.
3. Check that your systems have sufficient disk space for the installation. See Disk space requirements.
4. Check that you have the correct licenses. See "License requirements" on page 194 and IBM MQ license information.

What to do next

When you have completed these tasks, you are ready to start preparing your system for installation. For the next steps in installing IBM MQ, see "Preparing the system on Linux" on page 333.

Related concepts:

"IBM MQ installation overview" on page 192
An overview of concepts and considerations for installing IBM MQ, with links to instructions on how to install, verify, and uninstall IBM MQ on each of the supported platforms.

Related information:

IBM MQ maintenance tasks

Hardware and software requirements on Linux systems

Linux

Before you install IBM MQ , check that your system meets the hardware and operating system software requirements for the particular components you intend to install.

For hardware and software requirements, see System Requirements for IBM MQ.

IBM MQ does not support host names that contain spaces. If you install IBM MQ on a system with a host name that contains spaces, you are unable to create any queue managers.

Java Message Service and SOAP transport

If you want to use Java Message Service and SOAP support, you need an IBM Java 7 SDK and Runtime Environment Version 7.0 or later.

V 9.0.0 Java 8 is bundled with IBM MQ Version 9.0 but client components are built with Java 7 compatibility flags on.

For development, a JDK is required, and a JRE is required for running. The JRE does not need to be the JRE installed with IBM MQ, but has to be one from the supported list.

For a list of supported JDKs, see System Requirements for IBM MQ.

On Linux : Apache Axis V1.4 provides support for SOAP and is shipped on the server DVD, but not installed.

For further information about SOAP with IBM MQ , see IBM MQ transport for SOAP.

On Linux: On the Power platform, the 32-bit and 64-bit JDKs are typically installed to different locations, for example, the 32-bit JDK is located in `/opt/IBMJava2-ppc-50` and the 64-bit JDK is located in `/opt/IBMJava2-ppc64-50`. Ensure that the PATH variable is correctly set for your applications that use Java. To use the Postcard application described in "Verifying a local server installation using the Postcard application on Linux" on page 380, you must use a 32-bit JDK.

You can check the version installed using the following command:

```
java -version
```

Transport Layer Security (TLS)

If you want to use the TLS support, you need the IBM Global Security Kit (GSKit) V8 package. This package is supplied with IBM MQ as one of the components available for installation.

Linux

Installing the g++ version runtime support
If you intend to run TLS channels then you must have the g++ runtime libraries installed.

The GNU g++ libraries are called `libgcc_s.so` and `libstdc++.so.6`. On RPM based systems these are installed as part of the `libgcc` and `libstdc++` software packages.

The version of these libraries installed must be compatible with g++ version 3.4.

See System Requirements for IBM MQ for further details on the required packages for TLS support.

On 64 bit platforms, install both the 32 bit and the 64 bit versions of the package so that 32 bit and 64 bit processes can both use TLS functions.

IBM MQ Explorer requirements

Linux IBM MQ Explorer can be installed either as part of the product installation, or from the stand-alone IBM MQ Explorer support pack MS0T. See IBM MQ Explorer Requirements for the minimum requirements that your system needs, if you want to use the IBM MQ Explorer .

Note that IBM MQ Explorer is available for use only with IBM MQ for Linux, x86 and x86-64 platforms.

V 9.0.4

RDQM (replicated data queue manager)

Pacemaker is one of the prerequisites for RDQM. Pacemaker requires that the following Linux packages are installed on the system:

- OpenIPMI-libs.x86_64
- OpenIPMI-modalias.x86_64
- PyYAML.x86_64
- libesmtp.x86_64
- libyaml.x86_64
- net-snmp-agent-libs.x86_64
- openhpi-libs.x86_64

Related concepts:

IBM i “Hardware and software requirements on IBM i systems” on page 299
Check that the server environment meets the prerequisites for installing IBM MQ for IBM i. Check the product readme files and install missing prerequisite software supplied on the server CD.

“Hardware and software requirements on Windows systems” on page 445
Check that the server environment meets the prerequisites for installing IBM MQ for Windows and install any prerequisite software that is missing from your system from the server DVD.

Related tasks:

“Checking requirements on Windows” on page 444
Before you install IBM MQ on Windows, you must check for the latest information and system requirements.

Planning to install IBM MQ on Linux

Linux

Before you install IBM MQ on Linux, you must choose which components to install and where to install them. You must also make some platform-specific choices.

About this task

The following steps provide links to additional information to help you with planning your installation of IBM MQ on Linux.

As part of your planning activities, make sure that you review the information on hardware and software requirements for the platform on which you are planning to install IBM MQ. For more information, see "Checking requirements on Linux" on page 330.

Procedure

1. Decide which IBM MQ components and features to install. See "IBM MQ components and features" on page 192.

   Important: Ensure that your enterprise has the correct license, or licenses, for the components that you are going to install. For more information, see "License requirements" on page 194 and IBM MQ license information.
2. Review the options for naming your installation. In some cases, you can choose an installation name to use instead of the default name. See "Installation name on UNIX, Linux, and Windows" on page 197.
3. Review the options and restrictions for choosing an installation location for IBM MQ. For more information, see "Installation location on Multiplatforms" on page 198.
4. If you plan to install multiple copies of IBM MQ, see "Multiple installations on UNIX, Linux, and Windows" on page 200.
5. If you already have a primary installation, or plan to have one, see "Primary installation on UNIX, Linux, and Windows" on page 202.
6. Make sure that the communications protocol needed for server-to-server verification is installed and configured on both systems that you plan to use. For more information, see "Server-to-server links on UNIX, Linux, and Windows" on page 210.

Preparing the system on Linux

Linux

On Linux systems, you might have to complete several tasks before you install IBM MQ. You might also want to complete other tasks, depending on your installation intentions.

About this task

The tasks that you perform to prepare your systems for installation are listed here. Complete the appropriate tasks for your platform before installing.

Procedure

1. Set up a user ID of the name `mqm`, with a primary group of `mqm`. See "Setting up the user and group on Linux" on page 334.
2. Create file systems for both the product code and working data to be stored. See "Creating file systems on Linux" on page 335.
3. Configure any additional settings needed for your Linux system. See "Configuring and tuning the operating system on Linux" on page 337.

What to do next

When you have completed the tasks to prepare the system, you are ready to start installing IBM MQ. To install a server using rpm , see "Installing IBM MQ server on Linux" on page 344. To install a client using rpm, see "Installing an IBM MQ client on Linux" on page 355.

V 9.0.2 To install a server using a Debian installer, see "Installing an IBM MQ server on Linux Ubuntu using Debian packages" on page 368. To install a client using a Debian installer, see "Installing an IBM MQ client on Linux Ubuntu using Debian packages" on page 372

V 9.0.2

Important: Having both Debian and rpm installed versions of IBM MQ on the same system is not supported.

Related information:

Planning

Maintaining and migrating

IBM MQ maintenance tasks

Setting up the user and group on Linux

Linux

On Linux systems, IBM MQ requires a user ID of the name `mqm`, with a primary group of `mqm`. The `mqm` user ID owns the directories and files that contain the resources associated with the product.

Creating the user ID and group on UNIX and Linux systems

Set the primary group of the `mqm` user to the group `mqm`.

If you are installing IBM MQ on multiple systems you might want to ensure each UID and GID of `mqm` has the same value on all systems. If you are planning to configure multi-instance queue managers, it is essential the UID and GID are the same from system to system. It is also important to have the same UID and GID values in virtualization scenarios.

Linux RPM creates the `mqm` user ID and group ID as part of the installation procedure if they do not exist.

If you have special requirements for these IDs (for example they need to have the same values as other machines you are using, or your users and group ID are centrally managed) you should create the IDs before running the installation procedure, using the **`groupadd`** and **`useradd`** commands to set the UID and GID the same on each machine.

Note: The only IBM MQ requirement, is that the `mqm` user should have the `mqm` group as its primary group.

Adding existing user IDs to the group on Linux systems

If you want to run administration commands, for example **`crtmqm`** (create queue manager) or **`strmqm`** (start queue manager), your user ID must be a member of the `mqm` group. This user ID must not be longer than 12 characters.

Users do not need `mqm` group authority to run applications that use the queue manager; it is needed only for the administration commands.

Log files created by MQ Telemetry service

The **`umask`** setting of the user ID that creates a queue manager will determine the permissions of the Telemetry log files generated for that queue manager. Even though the ownership of the log files will be set to `mqm`.

Related concepts:

"Creating file systems on AIX" on page 223
Before installing IBM MQ, you might need to create file systems for both the product code and working data to be stored. There are minimum storage requirements for these file systems. The default installation directory for the product code can be changed at installation time, but the working data location cannot be changed.

"Configuring and tuning the operating system on HP-UX" on page 271
Before you install IBM MQ on an HP-UX system, you must check that the kernel is configured correctly.

"Configuring and tuning the operating system on Linux" on page 337
Use this topic when you are configuring IBM MQ on Linux systems.

Related tasks:

"Configuring and tuning the operating system on AIX" on page 224
When installing IBM MQ on AIX systems, there are some additional settings that must be configured.

Related information:

"Configuring and tuning the operating system on Solaris" on page 409
Configure Solaris systems with the resource limits required by IBM MQ.

Creating file systems on Linux

Linux

Before installing IBM MQ, you might need to create file systems for both the product code and working data to be stored. There are minimum storage requirements for these file systems. The default installation directory for the product code can be changed at installation time, but the working data location cannot be changed.

Determining the size of a server installations file system

To determine the size of the `/var/mqm` file system for a server installation, consider:

- The maximum number of messages in the system at one time.
- Contingency for message buildups, if there is a system problem.
- The average size of the message data, plus 500 bytes for the message header.
- The number of queues.
- The size of log files and error messages.
- The amount of trace that is written to the `/var/mqm/trace` directory.

Storage requirements for IBM MQ also depend on which components you install, and how much working space you need. For more details, see Disk space requirements.

Creating a file system for the working data

Before you install IBM MQ, create and mount a file system called `/var/mqm` which is owned by the user `mqm` in the group `mqm` ; see "Setting up the user and group on Linux" on page 334. This file system is used by all installations of IBM MQ on a system. If possible, use a partition strategy with a separate volume for the IBM MQ data. This means that other system activity is not affected if a large amount of IBM MQ work builds up. Configure the directory permissions to permit the `mqm` user to have full control, for example, file mode 755. These permissions will then be updated during the IBM MQ installation to match the permissions required by the queue manager.

Creating separate file systems for errors and logs

You can also create separate file systems for your log data (`/var/mqm/log`) and error files (`/var/mqm/errors`). If possible, place these directories on different physical disks from the queue manager data (`/var/mqm/qmgrs`) and from each other.

If you create separate file systems the `/var/mqm/errors` directory can be NFS mounted. However, if you choose to NFS-mount `/var/mqm/errors`, the error logs might be lost if the network fails.

You can protect the stability of your queue manager by having separate file systems for:
- `/var/mqm/errors`
- `/var/mqm/trace`
- `/var/mqm/qmgrs`
- `/var/mqm/log`

In the case of `/var/mqm/errors`, it is rare that this directory receives large quantities of data. But it is sometimes seen, particularly if there is a severe system problem leading to IBM MQ writing a lot of diagnostic information in to `.FDC` files. In the case of `/var/mqm/trace`, files are only written here when you use **`strmqtrc`** to start tracing IBM MQ.

You can obtain better performance of normal IBM MQ operations (for example, syncpoints, MQPUT, MQGET of persistent messages) by placing the following on separate disks:
- `/var/mqm/qmgrs`
- `/var/mqm/log`

In the rare event that you need to trace an IBM MQ system for problem determination, you can reduce performance impact by placing the `/var/mqm/trace` file system on a separate disk.

If you are creating separate file systems, allow a minimum of 30 MB of storage for `/var/mqm`, 100 MB of storage for `/var/mqm/log`, and 10 MB of storage for `/var/mqm/errors`. The 100 MB minimum allowance of storage for `/var/mqm/log` is the absolute minimum required for a single queue manager and is not a recommended value. The size of a file system must be scaled according to the number of queue managers that you intend to use, the number of pages per log file, and the number of log files per queue manager.

If you want to use individual queues that hold more than 2 GB of data, you must enable `/var/mqm` to use large files.

For more information about file systems, see File system support.

The size of the log file depends on the log settings that you use. The minimum sizes are for circular logging using the default settings. For more information about log sizes, see Calculating the size of the log.

Linux For a client installation, the file system can be mounted on a remote network device, for example NFS.

If you are performing both a client and a server installation, the requirements of the server installation take precedence over the requirements of the client installation.

Allow 15 MB as a minimum for an IBM MQ client.

A new sample IBM MQ MQI client configuration file is created in the `var/mqm` directory, by the client package, during installation, but only if this file does not exist. This file contains the `ClientExitPath` stanza. An example `mqclient.ini` file is shown in Configuring a client using a configuration file.

If you are using a common configuration file for multiple clients, either in the IBM MQ installation directory or in another location using the MQCLNTCF environment variable, you must grant read access to all user identifiers under which the IBM MQ client applications run. If, for any reason, the file cannot be read the failure is traced, and the search logic continues as if the file had not existed.

Related concepts:

"Setting up the user and group on Linux" on page 334
On Linux systems, IBM MQ requires a user ID of the name `mqm`, with a primary group of `mqm`. The `mqm` user ID owns the directories and files that contain the resources associated with the product.

"Configuring and tuning the operating system on Linux"
Use this topic when you are configuring IBM MQ on Linux systems.

Configuring and tuning the operating system on Linux

Linux

Use this topic when you are configuring IBM MQ on Linux systems.

Attention: The information in this topic applies only if the queue manager is started by the `mqm` user ID.

If any other user ID starts the queue manager, ensure that the **`NOFILE`** and **`NPROC`** entries, shown for `mqm`, are duplicated for that user ID.

Shell interpreter

Ensure that `/bin/sh` shell is a valid shell interpreter compatible with the Bourne shell, otherwise the post-installation configuration of IBM MQ does not complete successfully. If the shell was not installed using RPM, you might see a prerequisites failure of `/bin/sh` shell when you try to install IBM MQ . The failure is because the RPM tables do not recognize that a valid shell interpreter is installed. If the failure occurs, you can reinstall the `/bin/sh` shell by using RPM, or specify the RPM option `--nodeps` to disable dependency checking during installation of IBM MQ .

Note: The **`--dbpath`** option is not supported when installing IBM MQ on Linux.

System V IPC kernel configuration

IBM MQ uses System V IPC resources, in particular shared memory. However, a limited number of semaphores are also used.

The minimum configuration for IBM MQ for these resources is as follows:

Table 46. Minimum tunable kernel parameters values

Name	Kernel-name	Value	Increase	Description
`shmmni`	`kernel.shmmni`	4096	Yes	Maximum number of shared memory segments
`shmmax`	`kernel.shmmax`	268435456	No	Maximum size of a shared-memory segment (bytes)
`shmall`	`kernel.shmall`	2097152	Yes	Maximum amount of shared memory (pages)
`semmsl`	`kernel.sem`	32	No	Maximum amount of semaphores permitted per set
`semmns`	`kernel.sem`	4096	Yes	Maximum number of semaphores
`semopm`	`kernel.sem`	32	No	Maximum number of operations in single operations
`semmni`	`kernel.sem`	128	Yes	Maximum number of semaphore sets
`thrmax`	`kernel.threads-max`	32768	Yes	Maximum number of threads
`pidmax`	`kernel.pid_max`	32768	Yes	Maximum number of process identifiers

Notes:

1. These values are sufficient to run two moderate sized queue managers on the system. If you intend to run more than two queue managers, or the queue managers are to process a significant workload, you might need to increase the values displayed as Yes in the Increase column.
2. The `kernel.sem` values are contained within a single kernel parameter containing the four values in order.

To view the current value of the parameter log on, as a user with root authority, and type:

```
sysctl Kernel-name
```

To add or alter these values, log on as a user with root authority. Open the file `/etc/sysctl.conf` with a text editor, then add or change the following entries to your chosen values:

```
kernel.shmmni = 4096
kernel.shmall = 2097152
kernel.shmmax = 268435456
kernel.sem = 32 4096 32 128
```

Then save and close the file.

To load these **`sysctl`** values immediately, enter the following command `sysctl -p`.

If you do not issue the `sysctl -p` command, the new values are loaded when the system is rebooted.

By default the Linux kernel has a maximum process identifier, that can also be used with threads, and might limit the allowed number of threads.

The operating system reports when the system lacks the necessary resources to create another thread, or the system-imposed limit on the total number of threads in a process {PTHREAD_THREADS_MAX} would be exceeded.

For more information on `kernel.threads-max` and `kernel.pid-max`, see Resource shortage in IBM MQ queue manager when running a large number of clients

TCP/IP configuration

If you want to use **`keepalive`** for IBM MQ channels, you can configure the operation of the KEEPALIVE using the kernel parameters:

```
net.ipv4.tcp_keepalive_intvl
net.ipv4.tcp_keepalive_probes
net.ipv4.tcp_keepalive_time
```

See Using the TCP/IP SO_KEEPALIVE option for further information.

To view the current value of the parameter log on, as a user with root authority, and type `sysctl` *`Kernel-name`*.

To add or alter these values, log on as a user with root authority. Open the file `/etc/sysctl.conf` with a text editor, then add or change the following entries to your chosen values.

To load these **`sysctl`** values immediately, enter the following command `sysctl -p`.

If you do not issue the `sysctl -p` command, the new values are loaded when the system is rebooted.

Maximum open files

The maximum number of open file-handles in the system is controlled by the parameter **`fs.file-max`**

The minimum value for this parameter for a system with two moderate sized queue managers is 524288.

If you intend to run more than two queue managers, or the queue managers are to process a significant workload, you might need to increase this value.

To view the current value of a parameter, log on as a user with root authority, and type `sysctl fs.file-max`.

To add or alter these values, log on as a user with root authority. Open the file `/etc/sysctl.conf` with a text editor, then add or change the following entry to your chosen value:

```
fs.file-max = 524288
```

Then save and close the file.

To load these **`sysctl`** values immediately, enter the following command `sysctl -p`.

If you do not issue the `sysctl -p` command, the new values are loaded when the system is rebooted.

If you are using a pluggable security module such as PAM (Pluggable Authentication Module), ensure that this module does not unduly restrict the number of open files for the `mqm` user. To report the maximum number of open file descriptors per process for the `mqm` user, login as the `mqm` user and enter the following values:

```
ulimit -n
```

For a standard IBM MQ queue manager, set the *nofile* value for the `mqm` user to 10240 or more. To set the maximum number of open file descriptors for processes running under the `mqm` user, add the following information to the `/etc/security/limits.conf` file:

```
mqm       hard  nofile     10240
mqm       soft  nofile     10240
```

Maximum processes

A running IBM MQ queue manager consists of a number of thread programs. Each connected application increases the number of threads running in the queue manager processes. It is normal for an operating system to limit the maximum number of processes that a user runs. The limit prevents operating system failures due to an individual user or subsystem creating too many processes. You must ensure that the maximum number of processes that the `mqm` user is allowed to run is sufficient. The number of processes must include the number of channels and applications that connect to the queue manager.

The following calculation is useful when determining the number of processes for the `mqm` user:

```
nproc = 2048 + clientConnections * 4 + qmgrChannels * 4 +
    localBindingConnections
```

where:
- *clientConnections* is the maximum number of connections from clients on other machines connecting to queue managers on this machine.
- *qmgrChannels* is the maximum number of running channels (as opposed to channel definitions) to other queue managers. This includes cluster channels, sender/receiver channels, and so on.
- *localBindingConnections* does not include application threads.

The following assumptions are made in this algorithm:

- 2048 is a large enough contingency to cover the queue manager threads. This might need to be increased if a lot of other applications are running.
- When settting nproc, take into account the maximum number of applications, connections, channels and queue managers that might be run on the machine in the future.
- This algorithm takes a pessimistic view and the actual nproc needed might be slightly lower for later versions of IBM MQ and fastpath channels.
- V 8.0.0.1 V 8.0.2 In Linux, each thread is implemented as a light-weight process (LWP) and each LWP is counted as one process against nproc.

You can use the `PAM_limits` security module to control the number of processes that users run. You can configure the maximum number of processes for the `mqm` user as follows:

```
mqm       hard  nproc      4096
mqm       soft  nproc      4096
```

For more details on how to configure the `PAM_limits` security module type, enter the following command:

```
man limits.conf
```

You can check your system configuration using the mqconfig command.

For more information on configuring your system, see How to configure UNIX and Linux systems for IBM MQ.

Related concepts:

"Setting up the user and group on Linux" on page 334
On Linux systems, IBM MQ requires a user ID of the name `mqm`, with a primary group of `mqm`. The `mqm` user ID owns the directories and files that contain the resources associated with the product.

"Creating file systems on Linux" on page 335
Before installing IBM MQ, you might need to create file systems for both the product code and working data to be stored. There are minimum storage requirements for these file systems. The default installation directory for the product code can be changed at installation time, but the working data location cannot be changed.

Related information:

mqconfig

Installing IBM MQ on Linux using rpm

Linux

Installation tasks that are associated with installing IBM MQ on Linux systems using rpm are grouped in this section.

About this task

To install IBM MQ using rpm, complete the following tasks.

For information about how to uninstall IBM MQ, see "Uninstalling or modifying IBM MQ on Linux using rpm" on page 395.

If product fixes or updates are made available, see IBM MQ maintenance tasks for information about how to apply these changes.

Procedure

1. Check the system requirements. See "Checking requirements on Linux" on page 330.
2. Plan your installation.
   - As part of the planning process, you must choose which components to install and where to install them. See "IBM MQ rpm components for Linux systems."
   - You must also make some platform-specific choices. See "Planning to install IBM MQ on Linux" on page 332.
3. Prepare your system for installation of IBM MQ. See "Preparing the system on Linux" on page 333.
4. Install IBM MQ server. See "Installing IBM MQ server on Linux" on page 344.
5. Optional: Install an IBM MQ client. See "Installing an IBM MQ client on Linux" on page 355.
6. Verify your installation. See "Verifying an IBM MQ installation on Linux" on page 377.

IBM MQ rpm components for Linux systems

Linux

You can select the components that you require when you install IBM MQ.

Important: See IBM MQ license information for details of what each purchase of IBM MQ entitles you to install.

To display these components you can use, for example, the following command:

```
rpm -qa | grep MQ | xargs rpm -q --info
```

Table 47 shows the components that are available when installing an IBM MQ server or client on a Linux system:

Table 47. IBM MQ components for Linux systems

Component	Description	Server media	Client media	RPM package name
Runtime	Contains files that are common to both server and client installations. **Note:** MQSeriesRuntime component must be installed.	✓	✓	MQSeriesRuntime
Server	You can use the server to run queue managers on your system and connect to other systems over a network. Provides messaging and queuing services to applications, and support for IBM MQ client connections.	✓		MQSeriesServer
Standard Client	The IBM MQ MQI client is a small subset of IBM MQ, without a queue manager, that uses the queue manager and queues on other (server) systems. It can be used only when the system it is on is connected to another system that is running a full server version of IBM MQ. The client and the server can be on the same system if required.	✓	✓	MQSeriesClient
SDK	The SDK is required for compiling applications. It includes sample source files, and the bindings (files .H, .LIB, .DLL, and others), that you need to develop applications to run on IBM MQ.	✓	✓	MQSeriesSDK

Table 47. IBM MQ components for Linux systems (continued)

Component	Description	Server media	Client media	RPM package name
Sample programs	The sample application programs are needed if you want to check your IBM MQ installation using the verification procedures.	✓	✓	MQSeriesSamples
Java messaging	The files needed for messaging using Java (includes Java Message Service).	✓	✓	MQSeriesJava
Man pages	UNIX man pages, in U.S. English, for: control commands MQI calls MQSC commands	✓	✓	MQSeriesMan
Java JRE	A Java Runtime Environment that is used by those parts of IBM MQ that are written in Java.	✓	✓	MQSeriesJRE
Message Catalogs	For available languages, see the table of message catalogs that follows.	✓	✓	
IBM Global Security Kit	IBM Global Security Kit V8 Certificate and TLS, Base Runtime.	✓	✓	MQSeriesGSKit
Telemetry Service	MQ Telemetry supports the connection of Internet Of Things (IOT) devices (that is, remote sensors, actuators and telemetry devices) that use the IBM MQ Telemetry Transport (MQTT) protocol. The telemetry service, which is also know as the MQXR service, enables a queue manager to act as an MQTT server, and communicate with MQTT client apps. The telemetry service is only available on Linux for System x (64 bit) and Linux for IBM Z . A set of MQTT client libraries is also available in the IBM Messaging Telemetry Clients SupportPac. These libraries help you write the MQTT client apps that IOT devices use to communicate with MQTT servers. See also "Installation considerations for MQ Telemetry" on page 538.	✓		MQSeriesXRService
IBM MQ Explorer	Use IBM MQ Explorer to administer and monitor resources on Linux x86-64 systems. Also available using a stand-alone installer from MS0T.	✓		MQSeriesExplorer
Managed File Transfer	MQ Managed File Transfer transfers files between systems in a managed and auditable way, regardless of file size or the operating systems used. For information about the function of each component, see Managed File Transfer product options.	✓		MQSeriesFTAgent MQSeriesFTBase MQSeriesFTLogger MQSeriesFTService MQSeriesFTTools

Table 47. IBM MQ components for Linux systems (continued)

Component	Description	Server media	Client media	RPM package name
Advanced Message Security	Provides a high level of protection for sensitive data flowing through the IBM MQ network, while not impacting the end applications. You must install this component on all IBM MQ installations that host queues you want to protect. You must install the IBM Global Security Kit component on any IBM MQ installation that is used by a program that puts or gets messages to or from a protected queue, unless you are using only Java client connections. You must install the **Java JRE** component to install this component.	✓		MQSeriesAMS
V 9.0.0 AMQP Service	Install this component to make AMQP channels available. AMQP channels support MQ Light APIs. You can use AMQP channels to give AMQP applications access to the enterprise-level messaging facilities provided by IBM MQ.	✓		MQSeriesAMQP
V 9.0.1 REST API and Console	Adds HTTP based administration for IBM MQ through the REST API and IBM MQ Console.	✓		MQSeriesWeb
V 9.0.2 IBM MQ Bridge to Salesforce	Install this component to configure the connections to Salesforce and IBM MQ, then run the **runmqsfb** command to subscribe to events from Salesforce and publish them to an IBM MQ network. **Note:** The IBM MQ Bridge to Salesforce is available only on Linux for System x (64 bit).	✓	✓	MQSeriesSFBridge
V 9.0.4 IBM MQ Bridge to blockchain	Install this component to configure the connections to your blockchain network and IBM MQ. You can then run the **runmqbcb** command to start the bridge and send queries and updates to, and receive responses from your blockchain network. The queue manager that the bridge connects to must be an IBM MQ Advanced queue manager. For more information, see What is IBM MQ Advanced?. **Note:** The IBM MQ Bridge to blockchain is available only on Linux for System x (64 bit). This component is not supported for use with Docker.	✓	✓	MQSeriesBCBridge

Table 47. IBM MQ components for Linux systems (continued)

Component	Description	Server media	Client media	RPM package name
V 9.0.4 RDQM (replicated data queue manager)	Install this component to make the replicated data queue manager high availability configuration available. See "Installing RDQM (replicated data queue managers)" on page 545 for more information. This component is available only on Linux for System x (64 bit), on RHEL 7.3 or later. **Note:** This component is not supported for use with Docker.	✓		MQSeriesRDQM

Table 48. IBM MQ message catalogs for Linux systems

Message catalog language	RPM package name
Brazilian Portuguese	MQSeriesMsg_pt
Czech	MQSeriesMsg_cs
French	MQSeriesMsg_fr
German	MQSeriesMsg_de
Hungarian	MQSeriesMsg_hu
Italian	MQSeriesMsg_it
Japanese	MQSeriesMsg_ja
Korean	MQSeriesMsg_ko
Polish	MQSeriesMsg_pl
Russian	MQSeriesMsg_ru
Spanish	MQSeriesMsg_es
Simplified Chinese	MQSeriesMsg_Zh_CN
Traditional Chinese	MQSeriesMsg_Zh_TW
U.S. English	not applicable

Related concepts:

"IBM MQ components and features" on page 192
You can select the components or features that you require when you install IBM MQ.

"Planning considerations for installation on Multiplatforms" on page 196
Before you install IBM MQ, you must choose which components to install and where to install them. You must also make some platform-specific choices.

Installing IBM MQ server on Linux

Linux

You can install an IBM MQ server on a 64-bit Linux system.

Before you begin

- **CAUTION:**
  The instructions in this topic do not apply to Linux Ubuntu or Linux on POWER Systems - Little Endian. For information about these platforms, see "Installing IBM MQ server on Linux Ubuntu or Linux on POWER Systems - Little Endian" on page 350.

- If you install a copy of IBM MQ server for Linux using Electronic Software Download, obtained from Passport Advantage, you need to decompress the `tar.gz` file by using the **`gunzip`** command:

  ```
  gunzip IBM_MQ_V9.0_TRIAL_FOR_LINUX_ML.tar.gz
  ```

  and extract the installation files from the tar file, by using the following command:

  ```
  tar -xvf IBM_MQ_V9.0_TRIAL_FOR_LINUX_ML.tar
  ```

  Important: You must use GNU tar (also known as `gtar`) to unpack any `tar` images.
- Before you start the installation procedure, ensure that you have completed the necessary steps outlined in "Preparing the system on Linux" on page 333.
- If this installation is not the first installation on the system, you must ensure that the **`crtmqpkg`** command can write to a temporary location. By default, the **`crtmqpkg`** command will write to the `/var/tmp` directory. To use a different location, you can set the *TMPDIR* environment variable before you run the **`crtmqpkg`** command.
- To run the **`crtmqpkg`** command used in this task, you must have the **`pax`** command or **`rpmbuild`** installed.

  Attention: **`pax`** and **`rpmbuild`** are not supplied as part of the product. You must obtain these from your Linux distribution supplier.

About this task

Install the server by using the RPM Package Manager installer to select the components you want to install. The components and package names are listed in "Installing IBM MQ server on Linux Ubuntu or Linux on POWER Systems - Little Endian" on page 350.

Attention: If you install the packages using the wildcard character, that is, using the command `rpm -ivh MQ*.rpm`, you should install the packages in the following order:

- MQSeriesRuntime
- MQSeriesJRE
- MQSeriesJava
- MQSeriesServer
- MQSeriesWeb
- MQSeriesFTBase
- MQSeriesFTAgent
- MQSeriesFTService
- MQSeriesFTLogger
- MQSeriesFTTools
- V 9.0.0 MQSeriesAMQP
- MQSeriesAMS
- MQSeriesXRService
- MQSeriesExplorer
- MQSeriesGSKit
- MQSeriesClient
- MQSeriesMan
- MQSeriesMsg
- MQSeriesSamples
- MQSeriesSDK
- V 9.0.2 MQSeriesSFBridge
- V 9.0.4 MQSeriesBCBridge

Procedure

1. Log in as `root`, or switch to the superuser by using the **su** command.
2. Set your current directory to the location of the installation file. The location might be the mount point of the server DVD, a network location, or a local file system directory.
3. You must accept the terms of the license agreement before you can proceed with the installation. To do this run the `mqlicense.sh` script:

   ```
   ./mqlicense.sh
   ```

   The license agreement is displayed in a language appropriate to your environment and you are prompted to accept or decline the terms of the license.

   If possible, `mqlicense.sh` opens an X-window to display the license.

   If you need the license to be presented as text in the current shell, which can be read by a screen reader, type the following command, `./mqlicense.sh -text_only`
4. If this installation is not the only installation of IBM MQ on the system, you must run the **crtmqpkg** command to create a unique set of packages to install on the system. To run the **crtmqpkg** command to run on Linux, you must install the **pax** command and **rpmbuild**, which is located in the rpm-build package.

   Note: The **crtmqpkg** command is required only if this is not the first installation of IBM MQ on the system. If you have earlier versions of IBM MQ installed on your system, then installing the latest version works correctly if you install it in a different location.

   To run the **crtmqpkg** command on a Linux system:

   a. Enter the following command:

```
./crtmqpkg suffix
```

where *suffix* is a name of your choosing that uniquely identifies the installation packages on the system. *suffix* is not the same as an installation name, although the names can be identical. *suffix* is limited to 16 characters in the ranges A-Z, a-z, and 0-9.

Note: This command creates a full copy of the installation packages in a temporary directory. By default, the temporary directory is located at /var/tmp. You must ensure that the system has enough free space before you run this command. To use a different location, you can set the *TMPDIR* environment variable before you run the **crtmqpkg** command. For example:

```
$ TMPDIR=/test ./crtmqpkg suffix
```

b. Set your current directory to the location specified when the **crtmqpkg** command operation completes successfully. This directory is a subdirectory of the /var/tmp/mq_rpms directory, in which the unique set of packages is created. The packages have the *suffix* value contained within the file name. For example, using a suffix of "1":

```
./crtmqpkg 1
```

means there is a subdirectory named /var/tmp/mq_rpms/1/x86_64.

The packages are renamed according to the subdirectory, for example:

```
From: MQSeriesRuntime-8.0.0-0.x86_64.rpm
To: MQSeriesRuntime-1-8.0.0-0.x86_64.rpm
```

5. Install IBM MQ. To support the running of a queue manager, you must install at least the the MQSeriesRuntime and the MQSeriesServer components.
   - To install to the default location, /opt/mqm, use the **rpm -ivh** command to install each component that you require.

     For example, to install the runtime and server components to the default location, use the following command:

     ```
     rpm -ivh MQSeriesRuntime-*.rpm MQSeriesServer-*.rpm
     ```

     To install all components that are available in your current location on the installation media to the default location, use the following command:

     ```
     rpm -ivh MQSeries*.rpm
     ```

     Important: The components that you need to install might not all be in the same folder on the installation media. Some components might be under the /Advanced folder. For more information about installing IBM MQ Advanced components, see "Installing IBM MQ Advanced for Multiplatforms" on page 519.
   - To install to a non-default location, use the **rpm --prefix** option. For each installation, all of the IBM MQ components that you require must be installed in the same location.

     The installation path specified must be either an empty directory, the root of an unused file system, or a path that does not exist. The length of the path is limited to 256 bytes and must not contain spaces.

     For example, enter the following installation path to install the runtime and server components to the /opt/customLocation directory on a 64-bit Linux system:

     ```
     rpm --prefix /opt/customLocation -ivh MQSeriesRuntime-*.rpm
     MQSeriesServer-*.rpm
     ```

Results

You installed IBM MQ on your Linux system.

What to do next

- If required, you can now set this installation to be the primary installation. Enter the following command at the command prompt:

```
MQ_INSTALLATION_PATH/bin/setmqinst -i -p MQ_INSTALLATION_PATH
```

where *MQ_INSTALLATION_PATH* represents the directory where IBM MQ is installed.

You can have only one primary installation on a system. If there is already a primary installation on the system, you must unset it before you can set another installation as the primary installation. For more information, see Changing the primary installation.

- You might want to set up the environment to work with this installation. You can use the **setmqenv** or **crtmqenv** command to set various environment variables for a particular installation of IBM MQ . For more information, see setmqenv and crtmqenv.
- If you want to confirm that the installation was successful, you can verify your installation. See "Verifying an IBM MQ installation on Linux" on page 377, for more information.
- Only a user with a UID that is a member of the **mqm** group can issue administration commands. If you want to enable users to issue administration commands, they must be added to the **mqm** group. For more information, see "Setting up the user and group on Linux" on page 334 and Authority to administer IBM MQ on UNIX, Linux, and Windows systems.

Related concepts:

"Multiple installations on UNIX, Linux, and Windows" on page 200
On UNIX, Linux, and Windows, it is possible to have more than one copy of IBM MQ on a system.

"Primary installation on UNIX, Linux, and Windows" on page 202
On systems that support multiple installations of IBM MQ (UNIX, Linux, and Windows), the primary installation is the one to which IBM MQ system-wide locations refer. Having a primary installation is optional, but convenient.

Related tasks:

"Uninstalling or modifying IBM MQ on Linux using rpm" on page 395
On Linux, you can uninstall the IBM MQ server or client using the **rpm** command. You can also modify an installation by removing selected packages (components) currently installed on your system.

Related information:

setmqinst

Changing the primary installation

Installing the IBM MQ server silently on Linux: Linux

You can carry out a non-interactive installation of the IBM MQ server. A non-interactive installation is also known as a silent, or unattended installation.

About this task

To install IBM MQ silently, accept the IBM MQ license in non-interactive mode and then follow the interactive installation procedure.

Procedure

1. Log in as `root`, or switch to the superuser by using the **su** command.
2. You must accept the terms of the license agreement before you can proceed with the installation. To do this run the `mqlicense.sh` script.

   The license agreement will be displayed in a language appropriate to your environment and you will be prompted to accept or decline the terms of the license.

   If possible, `mqlicense.sh` opens an X-window to display the license.

   If you need the license to be presented as text in the current shell, which can be read by a screen reader, type the following command, `mqlicense.sh -text_only`
3. Follow the procedure detailed in "Installing IBM MQ server on Linux" on page 344 or "Installing IBM MQ server on Linux Ubuntu or Linux on POWER Systems - Little Endian" on page 350 as appropriate.

Related concepts:

"Multiple installations on UNIX, Linux, and Windows" on page 200
On UNIX, Linux, and Windows, it is possible to have more than one copy of IBM MQ on a system.

"Primary installation on UNIX, Linux, and Windows" on page 202
On systems that support multiple installations of IBM MQ (UNIX, Linux, and Windows), the primary installation is the one to which IBM MQ system-wide locations refer. Having a primary installation is optional, but convenient.

Related tasks:

"Uninstalling or modifying IBM MQ on Linux using rpm" on page 395
On Linux, you can uninstall the IBM MQ server or client using the **rpm** command. You can also modify an installation by removing selected packages (components) currently installed on your system.

Related information:

setmqinst

Changing the primary installation

Checking the availability of rpm on your Linux Ubuntu machine: Linux

If you are using an rpm installer, you must ensure that rpm is installed on your Linux Ubuntu machine before installing IBM MQ. V 9.0.2 From IBM MQ Version 9.0.2 you can use a Debian installer as an alternative to rpm.

Before you begin

V 9.0.2 If you choose to use the Debian installer, see "Installing IBM MQ on Linux Ubuntu using Debian" on page 363.

About this task

Important: The installation procedure uses the same rpm packages that are used by the other rpm based distributions. Technologies that convert these rpm packages into other forms, such as alien to convert rpms to Debian packages, are not compatible with the IBM MQ rpm packages and must not be used.

Procedure

1. To determine if the correct rpm package is installed on you system use the following command:
   ```
   dpkg-query -W --showformat '${Status}\n' rpm
   ```
   If you receive a response that is of the form:
   ```
   install ok installed
   ```
   rpm is installed on your system and no further action is required.

   If you receive a response that is of the form:
   ```
   unknown ok not-installed
   ```
   rpm is not installed on your system and you must install the rpm package before attempting to install IBM MQ, using the command described in step 2.
2. Run the following command, using root authority. In the example, you obtain root authority using the `sudo` command:
   ```
   sudo apt-get install rpm
   ```
   Attention: If this command does not complete successfully, consult your system administrator for instructions specific to your system on how to install the rpm package.

What to do next

You are now ready to install IBM MQ.

Related concepts:

"Multiple installations on UNIX, Linux, and Windows" on page 200
On UNIX, Linux, and Windows, it is possible to have more than one copy of IBM MQ on a system.

"Primary installation on UNIX, Linux, and Windows" on page 202
On systems that support multiple installations of IBM MQ (UNIX, Linux, and Windows), the primary installation is the one to which IBM MQ system-wide locations refer. Having a primary installation is optional, but convenient.

Related tasks:

"Uninstalling or modifying IBM MQ on Linux using rpm" on page 395
On Linux, you can uninstall the IBM MQ server or client using the **rpm** command. You can also modify an installation by removing selected packages (components) currently installed on your system.

Related information:

setmqinst

Changing the primary installation

Installing IBM MQ server on Linux Ubuntu or Linux on POWER Systems - Little Endian:

Linux

You can install an IBM MQ server on a Linux Ubuntu system, or Linux on POWER Systems - Little Endian system, in accordance with the system requirements web page.

Before you begin

You can install IBM MQ on a Linux Ubuntu system by using the RPM installer or by using the Debian installer. For more information about using the Debian installer, see "Installing IBM MQ on Linux Ubuntu using Debian" on page 363.

See System Requirements for IBM MQ for details of the supported software levels.

- Before you start the installation procedure, make sure that you have completed the necessary steps outlined in "Preparing the system on Linux" on page 333.
- If you install a copy of IBM MQ server for Linux Ubuntu using Electronic Software Download, obtained from Passport Advantage, you need to decompress the `tar.gz` file, and extract the installation files from the tar file, by using the following command:

  ```
  tar -xvf   WS_MQ_V9.0_TRIAL_FOR_LINUXUbuntu_ML.tar
  ```

  Important: You must use GNU tar (also known as `gtar`) to unpack any `tar` images.
- Ensure that RPM is installed on your system, as RPM is not installed by default on this platform.

  To determine if the correct RPM package is installed on you system, see "Checking the availability of rpm on your Linux Ubuntu machine" on page 349.
- Once RPM is installed on your system, carry out the following procedure, as root:
  1. Create directory `/etc/rpm`
  2. Add file `/macros`, containing the following code, `%_dbpath /var/lib/rpm`, to the `/etc/rpm` directory.

  Attention: You should only set up a `/macros` file if you are not already using RPM, as the previous instruction changes the default system-wide RPM database.

About this task

Install the server by using the RPM Package Manager installer to select the components that you want to install. The components and package names are listed in “IBM MQ components and features” on page 192.

Procedure

1. Open a shell terminal and set your current directory to the location of the installation packages. The location might be the mount point of the server DVD, a network location, or a local file system directory. You must have root authority to run the following commands. You can do so by adding **sudo** before the following commands, or by changing to the root user in the shell with the **su** command.
2. Run the `mqlicense.sh` script. If you want to view a text-only version of the license, which can be read by a screen reader, type the following message:

   ```
   ./mqlicense.sh -text_only
   ```

   The license is displayed.

   You must accept the license agreement before you can proceed with the installation.
3. If this installation is not the first installation of IBM MQ on the system, you must run the **crtmqpkg** command to create a unique set of packages to install on the system. For the **crtmqpkg** command to run on Linux, you must install the **pax** command and **rpmbuild**, which is located in the rpm package.
   a. Enter the following command:

      ```
      ./crtmqpkg suffix
      ```

      where *suffix* is a name of your choosing, that uniquely identifies the installation packages on the system. *suffix* is not the same as an installation name, although the names can be identical. *suffix* is limited to 16 characters in the ranges A-Z, a-z, and 0-9.

      Note: This command creates a full copy of the installation packages in a temporary directory. By default, the temporary directory is located at `/var/tmp`. You must ensure that the system has enough free space before you run this command. To use a different location, you can set the *TMPDIR* environment variable before you run the **crtmqpkg** command. For example:

      ```
      TMPDIR=/test ./crtmqpkg
      ```
   b. Set your current directory to the location specified when the **crtmqpkg** command completes. This directory is a subdirectory of the `/var/tmp/mq_rpms` directory, in which the unique set of packages is created. The packages have the *suffix* value contained within the file name. In the following example, the suffix of "1" `./crtmqpkg 1` means that there is a subdirectory named `/var/tmp/mq_rpms/1/i386`.

      The packages are renamed according to the subdirectory, for example, on Linux for System x (64-bit):

      ```
      From: MQSeriesRuntime-8.0.0-0.x86_64.rpm
      To: MQSeriesRuntime_1-8.0.0-0.x86_64.rpm
      ```
4. Install IBM MQ. At a minimum, you must install the `MQSeriesRuntime` and the `MQSeriesServer` components.

   If you are installing a subset of components, you must ensure that any dependencies are first installed, as listed in Table 49 on page 352.

   An additional flag is required when installing on Ubuntu 14.04 on Linux on POWER Systems - Little Endian:

   - **--ignorearch**: You must include this option to prevent problems with some levels of rpm not recognizing the Linux on POWER Systems - Little Endian architecture

   An additional flag is required when installing on Linux Ubuntu:

- **--force-debian**: You must include this option to prevent warning messages from the version of RPM for your platform, which indicates that the RPM packages are not intended to be directly installed using RPM.

To install the IBM MQ Explorer on Linux Ubuntu (x86-64 only):

a. Install all the components that you want, except for the IBM MQ Explorer component.
b. Install the IBM MQ Explorer component with the **--nodeps** flag. If you do not include the **--nodeps** flag, the installation fails with a dependency error. The dependency error occurs because the GTK2 packages are not installed by RPM and therefore cannot be found as package dependencies.

Notes:

- To install to the default location, `/opt/mqm`, use the rpm **-ivh** command to install each component that you require.

  To install the runtime and server components to the default location on Ubuntu Linux for System x (64-bit), use the following command:

  ```
  rpm -ivh --force-debian MQSeriesRuntime-*.rpm MQSeriesServer-*.rpm
  ```

  To install the runtime and server components to the default location on Linux on POWER Systems - Little Endian, use the following command:

  ```
  rpm -ivh --ignorearch MQSeriesRuntime-*.rpm MQSeriesServer-*.rpm
  ```

  To install all components to the default location on Linux on POWER Systems - Little Endian use the following command:

  ```
  rpm -ivh --ignorearch MQSeries*.rpm
  ```

- To install to a nondefault location, use the **rpm --prefix** option. For each installation, all of the IBM MQ components that you require must be installed in the same location.

  The installation path specified must be, either an empty directory, the root of an unused file system, or a path that does not exist.

  Attention: The length of the path is limited to 256 bytes and must not contain spaces.

  For example, enter the following installation path to install the runtime and server components to the `/opt/customLocation` directory on Linux on POWER Systems - Little Endian:

  ```
  rpm --prefix /opt/customLocation -ivh --ignorearch
  MQSeriesRuntime-*.rpm MQSeriesServer-*.rpm
  ```

Table 49 lists all available packages on Ubuntu, together with all the associated dependencies.

To install and use the package listed in the *Package Name* column, you must also install the components listed in the *Package Dependencies* column.

Table 49. Package component dependencies

Package Name	Component Function	Dependencies
MQSeriesRuntime	Common function for all other components	None
MQSeriesServer	Queue Manager	MQSeriesRuntime
MQSeriesClient	C IBM MQ client libraries	MQSeriesRuntime
MQSeriesJava	Java and JMS IBM MQ APIs	MQSeriesRuntime
MQSeriesJRE	Java Runtime Environment	MQSeriesRuntime

Table 49. Package component dependencies (continued)

Package Name	Component Function	Dependencies
MQSeriesExplorer	IBM MQ Explorer IBM MQ Explorer is only available on Linux for System x (64-bit). There is no IBM support for this component on Ubuntu, unless you are running on Ubuntu Version 14.04 (or later) and have installed IBM MQ Version 8.0.0, Fix Pack 2 (or later).	MQSeriesRuntime MQSeriesJRE GTK2 version 2.2.4-0 or later, including the GTK2 engines that contain the GTK2 themes Bitstream-vera-fonts
MQSeriesGSKit	IBM Global Security Kit **Note:** There is no IBM support for this component on Ubuntu, unless you are running on Ubuntu Version 14.04 (or later) and have installed IBM MQ Version 8.0.0, Fix Pack 2 (or later).	MQSeriesRuntime MQSeriesJRE
MQSeriesWeb	REST API and IBM MQ Console.	MQSeriesRuntime MQSeriesServer MQSeriesJava MQSeriesJRE
MQSeriesSDK	Header files and libraries for non-Java APIs	MQSeriesRuntime
MQSeriesMan	UNIX man pages for IBM MQ	MQSeriesRuntime
MQSeriesSamples	IBM MQ application samples	MQSeriesRuntime
MQSeriesMsg_cz MQSeriesMsg_de MQSeriesMsg_es MQSeriesMsg_fr MQSeriesMsg_hu MQSeriesMsg_it MQSeriesMsg_ja MQSeriesMsg_ko MQSeriesMsg_pl MQSeriesMsg_pt MQSeriesMsg_ru MQSeriesMsg_Zh_CN MQSeriesMsg_Zh_TW	Additional language message catalog files. English message catalog files are installed by default. For more information about these message catalogs, see “Displaying messages in your national language on Linux” on page 376	MQSeriesRuntime
MQSeriesFTBase	Managed File Transfer component	MQSeriesRuntime MQSeriesJava MQSeriesJRE

Table 49. Package component dependencies (continued)

Package Name	Component Function	Dependencies
MQSeriesFTLogger	Managed File Transfer component	MQSeriesRuntime MQSeriesServer MQSeriesFTBase MQSeriesJava MQSeriesJRE
MQSeriesFTTools MQSeriesFTAgent	Managed File Transfer components	MQSeriesRuntime MQSeriesFTBase MQSeriesJava MQSeriesJRE
MQSeriesFTService	Managed File Transfer component	MQSeriesRuntime MQSeriesServer MQSeriesFTAgent MQSeriesFTBase MQSeriesJava MQSeriesJRE
MQSeriesAMS	Advanced Message Security component **Note:** There is no IBM support for this component on Ubuntu, unless you are running on Ubuntu Version 14.04 (or later) and have installed IBM MQ Version 8.0.0, Fix Pack 2 (or later).	MQSeriesRuntime MQSeriesServer
V 9.0.2 MQSeriesSFBridge	Install the IBM MQ Bridge to Salesforce to subscribe to Salesforce push topics and platform events. V 9.0.4 From IBM MQ Version 9.0.4 you can also use the bridge to create event messages for Salesforce platform events. **Note:** The IBM MQ Bridge to Salesforce is available only on Linux for System x (64 bit).	ibmmq-runtime ibmmq-java ibmmq-jre
MQSeriesBCBridge	Install the IBM MQ Bridge to blockchain to send queries and updates to, and receive responses from your blockchain network. **Note:** The IBM MQ Bridge to blockchain is available only on Linux for System x (64 bit).	ibmmq-runtime ibmmq-java ibmmq-jre

Results

You have installed the packages you require.

What to do next

- If you have chosen this installation to be the primary installation on the system, you must now set it as the primary installation. Enter the following command at the command prompt:

  ```
  MQ_INSTALLATION_PATH/bin/setmqinst -i -p MQ_INSTALLATION_PATH
  ```

  where *MQ_INSTALLATION_PATH* represents the directory where IBM MQ is installed.

  You can have only one primary installation on a system. If there is already a primary installation on the system, you must unset it before you can set another installation as the primary installation. For more information, see Changing the primary installation.
- You might want to set up the environment to work with this installation. You can use the **setmqenv** or **crtmqenv** command to set various environment variables for a particular installation of IBM MQ . For more information, see setmqenv and crtmqenv.
- If you want to confirm that the installation was successful, you can verify your installation. See "Verifying an IBM MQ installation on Linux" on page 377, for more information.

Related concepts:

"Multiple installations on UNIX, Linux, and Windows" on page 200
On UNIX, Linux, and Windows, it is possible to have more than one copy of IBM MQ on a system.

"Primary installation on UNIX, Linux, and Windows" on page 202
On systems that support multiple installations of IBM MQ (UNIX, Linux, and Windows), the primary installation is the one to which IBM MQ system-wide locations refer. Having a primary installation is optional, but convenient.

Related tasks:

"Uninstalling or modifying IBM MQ on Linux using rpm" on page 395
On Linux, you can uninstall the IBM MQ server or client using the **rpm** command. You can also modify an installation by removing selected packages (components) currently installed on your system.

Related information:

setmqinst

Changing the primary installation

Installing an IBM MQ client on Linux

Linux

Installing an IBM MQ client on a 32 bit or 64 bit Linux system.

Before you begin

- Before you start the installation procedure, make sure that you have completed the necessary steps outlined in "Preparing the system on Linux" on page 333.
- If this installation is not the only installation on the system, you must ensure that you have write access to `/var/tmp`.

About this task

This task describes the installation of the client, using the RPM Package Manager installer to select which components you want to install. You must install at least the Runtime and Client components. The components are listed in "IBM MQ rpm components for Linux systems" on page 341.

Procedure

1. Log in as root, or switch to the superuser using the **su** command.
2. Make your current directory the location of the installation file. The location might be the mount point of the DVD, a network location, or a local file system directory.
3. Run the `mqlicense.sh` script. If you want to view a text-only version of license, which can be read by a screen-reader, type:

   ```
   ./mqlicense.sh -text_only
   ```

   The license is displayed.

   If want to accept the license without it being displayed, you can run the `mqlicense.sh` script with the `-accept` option.

   ```
   ./mqlicense.sh -accept
   ```

   You must accept the license agreement before you can proceed with the installation.
4. If you have multiple installations on this system, you must run **crtmqpkg** to create a unique set of packages to install on the system:
   a. Enter the following command:

      ```
      ./crtmqpkg suffix
      ```

      where *suffix* is a name of your choosing, that will uniquely identify the installation packages on the system. *suffix* is not the same as an installation name, although the names can be identical. *suffix* is limited to 16 characters in the ranges A-Z, a-z, and 0-9.
   b. Set your current directory to the location specified when the **crtmqpkg** command completes. This directory is a sub-directory of `/var/tmp/mq_rpms`, in which the unique set of packages is created. The packages have the *suffix* value contained within the filename.
5. Install IBM MQ. The minimum components you must install are the MQSeriesRuntime and the MQSeriesClient.
   - To install to the default location, `/opt/mqm`, use the **rpm -ivh** command to install each component that you require.

     For example, to install all components to the default location use the following command:

     ```
     rpm -ivh MQSeries*.rpm
     ```

     If you are using Ubuntu, add the **--force-debian** attribute. For example, to install all components to the default location use the following command:

     ```
     rpm --force-debian -ivh MQSeries*.rpm
     ```

     You must include this option to prevent seeing warning messages from the version of RPM for your platform, which indicates that the RPM packages are not intended to be directly installed using RPM.
   - To install to a non-default location use the **rpm --prefix** option. For each installation, all of the IBM MQ components that you require must be installed in the same location.

     The installation path specified must either be an empty directory, the root of an unused file system, or a path that does not exist. The length of the path is limited to 256 bytes and must not contain spaces.

     For example, to install the runtime and server components to `/opt/customLocation` on a 64-bit Linux system:

```
rpm --prefix /opt/customLocation -ivh MQSeriesRuntime-V.R.M-F.x86_64.rpm
MQSeriesClient-V.R.M-F.x86_64.rpm
```

     where:

     V Represents the version of the product that you are installing

     R Represents the release of the product that you are installing

M Represents the modification of the product that you are installing

F Represents the fix pack level of the product that you are installing

What to do next

- If you have chosen this installation to be the primary installation on the system, you must now set it as the primary installation. Enter the following command at the command prompt:

  ```
  MQ_INSTALLATION_PATH/bin/setmqinst -i -p MQ_INSTALLATION_PATH
  ```

  You can have only one primary installation on a system. If there is already a primary installation on the system, you must unset it before you can set another installation as the primary installation. For more information, see Changing the primary installation.
- You might want to set up the environment to work with this installation. You can use the **setmqenv** or **crtmqenv** command to set various environment variables for a particular installation of IBM MQ. For more information, see setmqenv and crtmqenv.
- For instructions on how to verify your installation, see “Testing communication between a client and a server on Linux” on page 393

Related tasks:

“Uninstalling or modifying IBM MQ on Linux using rpm” on page 395
On Linux, you can uninstall the IBM MQ server or client using the **rpm** command. You can also modify an installation by removing selected packages (components) currently installed on your system.

Checking the availability of RPM on your machine: Linux

You must ensure that RPM is installed on your Linux machine before installing IBM MQ.

V 9.0.2 From IBM MQ Version 9.0.2 you can use a Debian installer as an alternative to rpm.

Before you begin

V 9.0.2 If you choose to use the Debian installer, see “Installing IBM MQ on Linux Ubuntu using Debian” on page 363.

About this task

Important: The installation procedure uses the same rpm packages that are used by the other rpm based distributions. Technologies that convert these rpm packages into other forms, such as alien to convert rpms to Debian packages, are not compatible with the IBM MQ rpm packages and must not be used.

Procedure

1. To determine if the correct rpm package is installed on you system use the following command:

   ```
   dpkg-query -W --showformat '${Status}\n' rpm
   ```

   If you receive a response that is of the form:

   ```
   install ok installed
   ```

   rpm is installed on your system and no further action is required.

   If you receive a response that is of the form:

   ```
   unknown ok not-installed
   ```

   rpm is not installed on your system and you must install the rpm package before attempting to install IBM MQ, using the command described in step 2 on page 349.
2. Run the following command, using root authority. In the example, you obtain root authority using the `sudo` command:

```
sudo apt-get install rpm
```

Attention: If this command does not complete successfully, consult your system administrator for instructions specific to your system on how to install the rpm package.

What to do next

You are now ready to install IBM MQ.

Related concepts:

"Multiple installations on UNIX, Linux, and Windows" on page 200
On UNIX, Linux, and Windows, it is possible to have more than one copy of IBM MQ on a system.

"Primary installation on UNIX, Linux, and Windows" on page 202
On systems that support multiple installations of IBM MQ (UNIX, Linux, and Windows), the primary installation is the one to which IBM MQ system-wide locations refer. Having a primary installation is optional, but convenient.

Related tasks:

"Uninstalling or modifying IBM MQ on Linux using rpm" on page 395
On Linux, you can uninstall the IBM MQ server or client using the **`rpm`** command. You can also modify an installation by removing selected packages (components) currently installed on your system.

Related information:

setmqinst

Changing the primary installation

Installing IBM MQ client on Linux Ubuntu or Linux on POWER Systems - Little Endian:

Linux

You can install an IBM MQ client on a Linux Ubuntu , or Linux on POWER Systems - Little Endian, system in accordance with the system requirements web page.

Before you begin

See System Requirements for IBM MQ for details of the supported software levels.

- Before you start the installation procedure, make sure that you have completed the necessary steps outlined in "Preparing the system on Linux" on page 333.
- Ensure that RPM is installed on your system, as RPM is not installed by default on this platform.

  To determine if the correct RPM package is installed on you system, see "Checking the availability of RPM on your machine" on page 357.
- Once RPM is installed on your system, carry out the following procedure, as root:
  1. Create directory `/etc/rpm`
  2. Add file `/macros`, containing the following code, `%_dbpath /var/lib/rpm`, to the `/etc/rpm` directory.

Attention: You should only set up a `/macros` file if you are not already using RPM, as the previous instruction changes the default system-wide RPM database.

About this task

Install the client by using the RPM Package Manager installer to select the components that you want to install. The components and package names are listed in "IBM MQ components and features" on page 192.

Procedure

1. Open a shell terminal and set your current directory to the location of the installation packages. The location might be the mount point of the client DVD, a network location, or a local file system directory. You must have root authority to run the following commands. You can do so by adding **sudo** before the following commands, or by changing to the root user in the shell with the **su** command.
2. Run the `mqlicense.sh` script. If you want to view a text-only version of the license, which can be read by a screen reader, type the following message:
   ```
   ./mqlicense.sh -text_only
   ```
   The license is displayed.

   You must accept the license agreement before you can proceed with the installation.
3. If this installation is not the first installation of IBM MQ on the system, you must run the **crtmqpkg** command to create a unique set of packages to install on the system. For the **crtmqpkg** command to run on Linux, you must install the **pax** command and **rpmbuild**, which is located in the rpm package.
   a. Enter the following command:
      ```
      ./crtmqpkg suffix
      ```
      where *suffix* is a name of your choosing, that uniquely identifies the installation packages on the system. *suffix* is not the same as an installation name, although the names can be identical. *suffix* is limited to 16 characters in the ranges A-Z, a-z, and 0-9.

      Note: This command creates a full copy of the installation packages in a temporary directory. By default, the temporary directory is located at `/var/tmp`. You must ensure that the system has enough free space before you run this command. To use a different location, you can set the *TMPDIR* environment variable before you run the **crtmqpkg** command. For example:
      ```
      TMPDIR=/test ./crtmqpkg
      ```
   b. Set your current directory to the location specified when the **crtmqpkg** command completes. This directory is a subdirectory of the `/var/tmp/mq_rpms` directory, in which the unique set of packages is created. The packages have the *suffix* value contained within the file name. In the following example, the suffix of "1" `./crtmqpkg 1` means that there is a subdirectory named `/var/tmp/mq_rpms/1/i386`.

      The packages are renamed according to the subdirectory, for example, on Linux for System x (64-bit):
      ```
      From: MQSeriesRuntime-8.0.0-0.x86_64.rpm
      To: MQSeriesRuntime_1-8.0.0-0.x86_64.rpm
      ```
4. Install IBM MQ. At a minimum, you must install the `MQSeriesRuntime` component.

   An additional flag is required when installing on Linux Ubuntu:
   - **--force-debian**: You must include this option to prevent warning messages from the version of RPM for your platform, which indicates that the RPM packages are not intended to be directly installed using RPM.

   An additional flag is required when installing on Ubuntu 14.04 on Linux on POWER Systems - Little Endian:
   - **--ignorearch**: You must include this option to prevent problems with some levels of rpm not recognizing the Linux on POWER Systems - Little Endian architecture

   If you are installing a subset of components, you must ensure that any dependencies are first installed, as listed in Table 50 on page 360.

   Notes:
   - To install to the default location, `/opt/mqm`, use the rpm **-ivh** command to install each component that you require.

To install the runtime component to the default location on Ubuntu Linux for System x (64-bit), use the following command:

```
rpm -ivh --force-debian MQSeriesRuntime-*.rpm
```

To install the runtime component to the default location on Ubuntu Linux on POWER Systems - Little Endian, use the following command:

```
rpm -ivh --force-debian --ignorearch MQSeriesRuntime-*.rpm
```

To install all components to the default location on Ubuntu Linux on POWER Systems - Little Endian use the following command:

```
rpm -ivh --force-debian --ignorearch MQSeries*.rpm
```

- To install to a nondefault location, use the **`rpm --prefix`** option. For each installation, all of the IBM MQ components that you require must be installed in the same location.

  The installation path specified must be, either an empty directory, the root of an unused file system, or a path that does not exist.

  Attention: The length of the path is limited to 256 bytes and must not contain spaces.

  For example, enter the following installation path to install the runtime component to the `/opt/customLocation` directory on Ubuntu Linux on POWER Systems - Little Endian:

  ```
  rpm --prefix /opt/customLocation -ivh --force-debian --ignorearch
  MQSeriesRuntime-*.rpm
  ```

Table 50 lists all available packages on Ubuntu, together with all the associated dependencies.

To install and use the package listed in the *Package Name* column, you must also install the components listed in the *Package Dependencies* column.

Table 50. Package component dependencies

Package Name	Component Function	Package Dependencies
MQSeriesRuntime	Common function for all other components	None
MQSeriesClient	C IBM MQ client libraries	MQSeriesRuntime
MQSeriesJava	Java and JMS IBM MQ APIs	MQSeriesRuntime
MQSeriesJRE	Java Runtime Environment	MQSeriesRuntime
MQSeriesExplorer	IBM MQ Explorer IBM MQ Explorer is only available on Linux for System x (64-bit). **Note:** There is no IBM support for this component on Ubuntu, unless you are running on Ubuntu Version 14.04 (or later) and have installed IBM MQ Version 8.0.0.2.	MQSeriesRuntime MQSeriesJRE GTK2 version 2.2.4-0 or later, including the GTK2 engines that contain the GTK2 themes Bitstream-vera-fonts
MQSeriesGSKit	IBM Global Security Kit **Note:** There is no IBM support for this component on Ubuntu, unless you are running on Ubuntu Version 14.04 (or later) and have installed IBM MQ Version 8.0.0.2.	MQSeriesRuntime MQSeriesJRE
MQSeriesSDK	Header files and libraries for non-Java APIs	MQSeriesRuntime
MQSeriesMan	UNIX man pages for IBM MQ	MQSeriesRuntime
MQSeriesSamples	IBM MQ application samples	MQSeriesRuntime

Table 50. Package component dependencies (continued)

Package Name	Component Function	Package Dependencies
MQSeriesMsg_cs MQSeriesMsg_de MQSeriesMsg_es MQSeriesMsg_fr MQSeriesMsg_hu MQSeriesMsg_it MQSeriesMsg_ja MQSeriesMsg_ko MQSeriesMsg_pl MQSeriesMsg_pt MQSeriesMsg_ru MQSeriesMsg_Zh_CN MQSeriesMsg_Zh_TW	Language specific message catalog files	MQSeriesRuntime
MQSeriesFTBase	Managed File Transfer component	MQSeriesRuntime MQSeriesJava MQSeriesJRE
MQSeriesFTLogger	Managed File Transfer component	MQSeriesRuntime MQSeriesFTBase MQSeriesJava MQSeriesJRE
MQSeriesFTTools MQSeriesFTAgent	Managed File Transfer components	MQSeriesRuntime MQSeriesFTBase MQSeriesJava MQSeriesJRE
MQSeriesFTService	Managed File Transfer component	MQSeriesRuntime MQSeriesFTAgent MQSeriesFTBase MQSeriesJava MQSeriesJRE
MQSeriesAMS	Advanced Message Security component **Note:** There is no IBM support for this component on Ubuntu, unless you are running on Ubuntu Version 14.04 (or later) and have installed IBM MQ Version 8.0.0.2.	MQSeriesRuntime

Table 50. Package component dependencies (continued)

Package Name	Component Function	Package Dependencies
V 9.0.2 MQSeriesSFBridge	Install the IBM MQ Bridge to Salesforce to subscribe to Salesforce push topics and platform events. V 9.0.4 From IBM MQ Version 9.0.4 you can also use the bridge to create event messages for Salesforce platform events. **Note:** The IBM MQ Bridge to Salesforce is available only on Linux for System x (64 bit).	ibmmq-runtime ibmmq-java ibmmq-jre
MQSeriesBCBridge	Install the IBM MQ Bridge to blockchain to send queries and updates to, and receive responses from your blockchain network. **Note:** The IBM MQ Bridge to blockchain is available only on Linux for System x (64 bit).	ibmmq-runtime ibmmq-java ibmmq-jre

Results

You have installed the packages you require.

What to do next

- If you have chosen this installation to be the primary installation on the system, you must now set it as the primary installation. Enter the following command at the command prompt:

  ```
  MQ_INSTALLATION_PATH/bin/setmqinst -i -p MQ_INSTALLATION_PATH
  ```

  You can have only one primary installation on a system. If there is already a primary installation on the system, you must unset it before you can set another installation as the primary installation. For more information, see Changing the primary installation.
- You might want to set up the environment to work with this installation. You can use the **setmqenv** or **crtmqenv** command to set various environment variables for a particular installation of IBM MQ. For more information, see setmqenv and crtmqenv.
- For instructions on how to verify your installation, see "Testing communication between a client and a server on Linux" on page 393

Related concepts:

"Multiple installations on UNIX, Linux, and Windows" on page 200
On UNIX, Linux, and Windows, it is possible to have more than one copy of IBM MQ on a system.

"Primary installation on UNIX, Linux, and Windows" on page 202
On systems that support multiple installations of IBM MQ (UNIX, Linux, and Windows), the primary installation is the one to which IBM MQ system-wide locations refer. Having a primary installation is optional, but convenient.

Related tasks:

"Uninstalling or modifying IBM MQ on Linux using rpm" on page 395
On Linux, you can uninstall the IBM MQ server or client using the **rpm** command. You can also modify an installation by removing selected packages (components) currently installed on your system.

Related information:

setmqinst

Changing the primary installation

Installing IBM MQ on Linux Ubuntu using Debian

Linux V 9.0.2

Installation tasks that are associated with installing IBM MQ on Linux systems using a Debian installer are grouped in this section.

Before you begin

Important: Using both Debian and rpm to install IBM MQ on the same system is not supported.

About this task

To install IBM MQ using a Debian installer, complete the following tasks.

If product fixes or updates are made available, see IBM MQ maintenance tasks for information about how to apply these changes.

Procedure

1. Check the system requirements. See "Checking requirements on Linux" on page 330.
2. Plan your installation. As part of the planning process, you must choose which components to install and where to install them. See "IBM MQ Debian components for Linux Ubuntu systems" on page 364.
3. Prepare your system for installation of IBM MQ. See "Preparing the system on Linux" on page 333.
4. Install IBM MQ server. See "Installing an IBM MQ server on Linux Ubuntu using Debian packages" on page 368.
5. Optional: Install an IBM MQ client. See "Installing an IBM MQ client on Linux Ubuntu using Debian packages" on page 372.
6. Verify your installation. See "Verifying an IBM MQ installation on Linux" on page 377.

Overview of the Debian installer for IBM MQ on Linux Ubuntu

▶ Linux ▶ V 9.0.2

An overview of the concepts and considerations for installing IBM MQ, on Linux Ubuntu, using the Debian installer.

Installation tools

Use **dpkg**, or a higher level installation tool, to install and uninstall the product. The installed product on disk appears identical to an rpm-installed copy.

Attention: The Debian installation tools have no provision for overriding the installation directory. This means that there is *no relocatable or multi-version support*. Therefore the product will be installed to `/opt/mqm`, but this can be set as the primary installation if you require.

Package names

The package names have been changed to use an IBM MQ derived name.

For example, the Debian equivalent of the existing rpm server component, `MQSeriesServer`, is `ibmmq-server`.

Coexistence

Having both Debian and rpm installed versions of IBM MQ on the same system is not supported.

Therefore, on a single system, you can have a single version of IBM MQ installed by Debian, or multiple versions of IBM MQ installed by rpm.

However, you can achieve multi-version installation with Debian through the use of container based technologies, such as Docker.

Related tasks:

"Installing IBM MQ server on Linux Ubuntu or Linux on POWER Systems - Little Endian" on page 350
You can install an IBM MQ server on a Linux Ubuntu system, or Linux on POWER Systems - Little Endian system, in accordance with the system requirements web page.

IBM MQ Debian components for Linux Ubuntu systems

▶ Linux ▶ V 9.0.2

You can select the components that you require when you install IBM MQ.

Table 51 shows the components that are available when installing an IBM MQ server or client on a Linux Ubuntu system using the Debian installer:

Table 51. IBM MQ Debian components for Linux Ubuntu systems

Component	Description	Server media	Client media	Debian package name
Runtime	Contains files that are common to both server and client installations. **Note:** ibmmq-runtime component must be installed.	✓	✓	ibmmq-runtime

Table 51. IBM MQ Debian components for Linux Ubuntu systems (continued)

Component	Description	Server media	Client media	Debian package name
Server	You can use the server to run queue managers on your system and connect to other systems over a network. Provides messaging and queuing services to applications, and support for IBM MQ client connections.	✓		ibmmq-server
Standard Client	The IBM MQ MQI client is a small subset of IBM MQ, without a queue manager, that uses the queue manager and queues on other (server) systems. It can be used only when the system it is on is connected to another system that is running a full server version of IBM MQ. The client and the server can be on the same system if required.	✓	✓	ibmmq-client
SDK	The SDK is required for compiling applications. It includes sample source files, and the bindings (files .H, .LIB, .DLL, and others), that you need to develop applications to run on IBM MQ.	✓	✓	ibmmq-sdk
Sample programs	The sample application programs are needed if you want to check your IBM MQ installation using the verification procedures.	✓	✓	ibmmq-samples
Java messaging	The files needed for messaging using Java (includes Java Message Service).	✓	✓	ibmmq-java
Man pages	UNIX man pages, in U.S. English, for: control commands MQI calls MQSC commands	✓	✓	ibmmq-man
Java JRE	A Java Runtime Environment that is used by those parts of IBM MQ that are written in Java.	✓	✓	ibmmq-jre
Message Catalogs	For available languages, see the table of message catalogs that follows.	✓	✓	
IBM Global Security Kit	IBM Global Security Kit V8 Certificate and TLS, Base Runtime.	✓	✓	ibmmq-gskit

Table 51. IBM MQ Debian components for Linux Ubuntu systems (continued)

Component	Description	Server media	Client media	Debian package name
Telemetry Service	MQ Telemetry supports the connection of Internet Of Things (IOT) devices (that is, remote sensors, actuators and telemetry devices) that use the IBM MQ Telemetry Transport (MQTT) protocol. The telemetry service, which is also know as the MQXR service, enables a queue manager to act as an MQTT server, and communicate with MQTT client apps. The telemetry service is only available on Linux for System x (64 bit) and Linux for IBM Z . A set of MQTT client libraries is also available in the IBM Messaging Telemetry Clients SupportPac. These libraries help you write the MQTT client apps that IOT devices use to communicate with MQTT servers. See also "Installation considerations for MQ Telemetry" on page 538.	✓		ibmmq-xrservice
IBM MQ Explorer	Use IBM MQ Explorer to administer and monitor resources on Linux x86-64 systems. Also available using a stand-alone installer from MS0T.	✓		ibmmq-explorer
Managed File Transfer	MQ Managed File Transfer transfers files between systems in a managed and auditable way, regardless of file size or the operating systems used. For information about the function of each component, see Managed File Transfer product options.	✓		ibmmq-ftagent ibmmq-ftbase ibmmq-ftlogger ibmmq-ftservice ibmmq-fttools
Advanced Message Security	Provides a high level of protection for sensitive data flowing through the IBM MQ network, while not impacting the end applications. You must install this component on all IBM MQ installations that host queues you want to protect. You must install the IBM Global Security Kit component on any IBM MQ installation that is used by a program that puts or gets messages to or from a protected queue, unless you are using only Java client connections. You must install the **Java JRE** component to install this component.	✓		ibmmq-ams
V 9.0.0 AMQP Service	Install this component to make AMQP channels available. AMQP channels support MQ Light APIs. You can use AMQP channels to give AMQP applications access to the enterprise-level messaging facilities provided by IBM MQ.	✓		ibmmq-amqp

Table 51. IBM MQ Debian components for Linux Ubuntu systems (continued)

Component	Description	Server media	Client media	Debian package name
REST API and Console	Adds HTTP based administration for IBM MQ through the REST API and IBM MQ Console.	✓		ibmmq-web
V 9.0.2 IBM MQ Bridge to Salesforce	Install the IBM MQ Bridge to Salesforce to configure the connections to Salesforce and IBM MQ, then run the command **runmqsfb** to subscribe to events from Salesforce and publish them to an IBM MQ network. **Note:** The IBM MQ Bridge to Salesforce is available only on Linux for System x (64 bit).	✓	✓	ibmmq-sfbridge
V 9.0.4 IBM MQ Bridge to blockchain	Install the IBM MQ Bridge to blockchain to send queries and updates to, and receive responses from your blockchain network. **Note:** The IBM MQ Bridge to blockchain is available only on Linux for System x (64 bit).	✓	✓	ibmmq-bcbridge

Table 52. IBM MQ message catalogs for Linux systems

Message catalog language	Component name
Brazilian Portuguese	ibmmq-msg-pt
Czech	ibmmq-msg-cs
French	ibmmq-msg-fr
German	ibmmq-msg-de
Hungarian	ibmmq-msg-hu
Italian	ibmmq-msg-it
Japanese	ibmmq-msg-ja
Korean	ibmmq-msg-ko
Polish	ibmmq-msg-pl
Russian	ibmmq-msg-ru
Spanish	ibmmq-msg-es
Simplified Chinese	ibmmq-msg-zh-cn
Traditional Chinese	ibmmq-msg-zh-tw
U.S. English	not applicable

Related concepts:
"IBM MQ components and features" on page 192
You can select the components or features that you require when you install IBM MQ.

Installing an IBM MQ server on Linux Ubuntu using Debian packages

Linux V 9.0.2

You can install an IBM MQ server on a Linux Ubuntu system, using a Debian installer, in accordance with the system requirements web page.

Before you begin

See System Requirements for IBM MQ for details of the supported software levels.

Before you start the installation procedure, make sure that you first complete the necessary steps that are outlined in "Preparing the system on Linux" on page 333.

If you have installed IBM MQ Version 9.0.2, or earlier, on Ubuntu using rpm, you must uninstall all rpm versions of the product before installing the Debian version of the product.

About this task

Install the server by using a Debian installer to select the components that you want to install. The components and package names are listed in "IBM MQ Debian components for Linux Ubuntu systems" on page 364.

You can use various installers. This topic describes the use of the **dpkg** and **apt** installers.

dpkg Install the packages that you need by following instructions in step 3. You can install multiple packages with the same command but take care to place the packages in the correct order as **dpkg** does not sort them according to dependency.

apt Install the packages that you need by following instructions in step 4 on page 369. The **apt** tool installs dependency packages for the package that you require. The **apt** management tool orders the **dpkg** commands.

Procedure

1. Open a shell terminal and set your current directory to the location of the installation packages. The location might be the mount point of the server DVD, a network location, or a local file system directory. You must have root authority to run the following commands. You can do so by adding **sudo** before the following commands, or by changing to the root user in the shell with the **su** command.
2. Run the `mqlicense.sh` script. If you want to view a text-only version of the license, which can be read by a screen reader, type the following message:
   ```
   ./mqlicense.sh -text_only
   ```
   The license is displayed.

   You must accept the license agreement before you can proceed with the installation.
3. Complete this step if you want to use the **dpkg** command and install packages individually. Issue the **dpkg** command for each IBM MQ package. For example, issue the following command:
   ```
   dpkg -i ibmmq-runtime_9.0.2.0_amd64.deb
   ```
   To support the running of a queue manager, you must install at least the `ibmmq-runtime` and the `ibmmq-server` components.

Important: You can specify multiple package files in the same command but unlike rpm, **dpkg** does not sort the package files into dependency order.

You must place the package file names in the following order when you issue your command:
- ibmmq-runtime
- ibmmq-jre
- ibmmq-java
- ibmmq-server
- ibmmq-web
- ibmmq-ftbase
- ibmmq-ftagent
- ibmmq-ftservice
- ibmmq-ftlogger
- ibmmq-fttools
- ibmmq-amqp
- ibmmq-ams
- ibmmq-xrservice
- ibmmq-explorer
- ibmmq-gskit
- ibmmq-client
- ibmmq-man
- ibmmq-msg_*language*
- ibmmq-samples
- ibmmq-sdk
- ibmmq-sfbridge
- ibmmq-bcbridge

If you are installing a subset of components by using **dpkg**, you must ensure that any dependencies are first installed, as listed in Table 53 on page 370.

4. Complete this step if you want to use the **apt** management tool to install the IBM MQ packages that you want along with their dependency packages. **apt** is a higher-level package management tool that is a front end to **dpkg**.

   Important: **apt** operations, unlike **dpkg**, are dependency aware, and automatically select and install the required packages. Therefore, the **apt** management tool orders the **dpkg** commands appropriately.

   apt is configured with a list of repositories that can include local directories. To add a local, or nfs-mounted, directory that contains the IBM MQ packages:

   a. Create a file with the suffix `.list`, for example, `IBM_MQ.list`, in the `/etc/apt/sources.list.d` directory. This file should contain a `deb` entry for the location of the directory that contains the IBM MQ packages.

      For example:

      ```
      # Local directory containing IBM MQ packages
      deb [trusted=yes] file:/var/tmp/mq ./
      ```

      Note: The inclusion of the `[trusted=yes]` statement (including the brackets) is optional and suppresses warnings and prompts during subsequent operations.

   b. Run the command **apt update** to add this directory, and the list of packages the directory contains, to the apt cache. You can now carry out various operations. For example, issuing the command:

      ```
      apt install "ibmmq-*"
      ```

installs the complete product, and issuing the command:

```
apt install ibmmq-server
```

selects and installs the server package and all its dependencies.

Attention: Do not run the `apt install ibmmq-*` command in the directory which holds the `.deb` files, unless you are using quotation characters in the shell.

If you are using tools such as aptitude or synaptic, the install packages can be found in the `misc\non-free` category.

To support the running of a queue manager, you must install at least the `ibmmq-runtime` and the `ibmmq-server` components.

If you are installing a subset of components, you must ensure that any dependencies are first installed, as listed in Table 53. To install and use a package listed in the *Package Name* column, you must also install the corresponding components that are listed in the *Package Dependencies* column.

Table 53. Package component dependencies

Package Name	Component Function	Package Dependencies
ibmmq-runtime	Common function for all other components	None
ibmmq-server	Queue Manager	ibmmq-runtime
ibmmq-client	C IBM MQ client libraries	ibmmq-runtime
ibmmq-java	Java and JMS IBM MQ APIs	ibmmq-runtime
ibmmq-jre	Java Runtime Environment	ibmmq-runtime
ibmmq-sdk	Header files and libraries for non-Java APIs	ibmmq-runtime
ibmmq-man	UNIX man pages for IBM MQ	ibmmq-runtime
ibmmq-samples	IBM MQ application samples	ibmmq-runtime
ibmmq-msg-cz ibmmq-msg-de ibmmq-msg-es ibmmq-msg-fr ibmmq-msg-hu ibmmq-msg-it ibmmq-msg-ja ibmmq-msg-ko ibmmq-msg-pl ibmmq-msg-pt ibmmq-msg-ru ibmmq-msg-zh-cn ibmmq-msg-zh-tw	Additional language message catalog files. English message catalog files are installed by default. For more information about these message catalogs, see "Displaying messages in your national language on Linux" on page 376	ibmmq-runtime
ibmmq-mqexplorer	IBM MQ Explorer. Only on Linux x86-64 systems.	ibmmq-runtime ibmmq-jre

Table 53. Package component dependencies (continued)

Package Name	**Component Function**	**Package Dependencies**
ibmmq-gskit	IBM Global Security Kit	ibmmq-runtime ibmmq-jre
ibmmq-web	REST API and IBM MQ Console.	ibmmq-runtime ibmmq-server ibmmq-java ibmmq-jre
ibmmq-ftbase	Managed File Transfer component	ibmmq-runtime ibmmq-java ibmmq-jre
ibmmq-ftlogger	Managed File Transfer component	ibmmq-runtime ibmmq-server ibmmq-ftbase ibmmq-java ibmmq-jre
ibmmq-fttools ibmmq-ftagent	Managed File Transfer components	ibmmq-runtime ibmmq-ftbase ibmmq-java ibmmq-jre
ibmmq-ftservice	Managed File Transfer component	ibmmq-runtime ibmmq-server ibmmq-ftagent ibmmq-ftbase ibmmq-java ibmmq-jre
ibmmq-ams	Advanced Message Security component	ibmmq-runtime ibmmq-server
V 9.0.2 ibmmq-sfb	Install the IBM MQ Bridge to Salesforce to configure the connections to Salesforce and IBM MQ, then run the command **`runmqsfb`** to subscribe to events from Salesforce and publish them to an IBM MQ network. **Note:** The IBM MQ Bridge to Salesforce is available only on Linux for System x (64 bit).	ibmmq-runtime ibmmq-java ibmmq-jre

Table 53. Package component dependencies (continued)

Package Name	Component Function	Package Dependencies
V 9.0.4 ibmmq-bcb	Install the IBM MQ Bridge to blockchain to send queries and updates to, and receive responses from your blockchain network. **Note:** The IBM MQ Bridge to blockchain is available only on Linux for System x (64 bit).	ibmmq-runtime ibmmq-java ibmmq-jre

Results

You have installed the packages you require.

What to do next

- If required, you can now set this installation to be the primary installation. Enter the following command at the command prompt:

  ```
  MQ_INSTALLATION_PATH/bin/setmqinst -i -p MQ_INSTALLATION_PATH
  ```

  where *MQ_INSTALLATION_PATH* represents the directory where IBM MQ is installed.
- You might want to set up the environment to work with this installation. You can use the **setmqenv** or **crtmqenv** command to set various environment variables for a particular installation of IBM MQ . For more information, see setmqenv and crtmqenv.
- If you want to confirm that the installation was successful, you can verify your installation. See "Verifying an IBM MQ installation on Linux" on page 377, for more information.

Related tasks:

"Uninstalling or modifying IBM MQ on Linux using rpm" on page 395
On Linux, you can uninstall the IBM MQ server or client using the **rpm** command. You can also modify an installation by removing selected packages (components) currently installed on your system.

Installing an IBM MQ client on Linux Ubuntu using Debian packages

Linux V 9.0.2

You can install an IBM MQ client on a Linux Ubuntu system, using a Debian package, in accordance with the system requirements web page.

Before you begin

See System Requirements for IBM MQ for details of the supported software levels.

Before you start the installation procedure, make sure that you have completed the necessary steps outlined in "Preparing the system on Linux" on page 333.

About this task

Install the client by using a Debian installer to select the components that you want to install. The components and package names are listed in "IBM MQ Debian components for Linux Ubuntu systems" on page 364.

Procedure

1. Open a shell terminal and set your current directory to the location of the installation packages. The location might be the mount point of the client DVD, a network location, or a local file system

directory. You must have root authority to run the following commands. You can do so by adding **sudo** before the following commands, or by changing to the root user in the shell with the **su** command.

2. Run the `mqlicense.sh` script. If you want to view a text-only version of the license, which can be read by a screen reader, type the following message:

   ```
   ./mqlicense.sh -text_only
   ```

   The license is displayed.

   You must accept the license agreement before you can proceed with the installation.

3. Install the IBM MQ client. You can use any Debian installer. "Installing an IBM MQ server on Linux Ubuntu using Debian packages" on page 368 describes the use of the **dpkg** and **apt** packages to install a server. At a minimum, you must install the `ibmmq-runtime` component.

   If you are installing a subset of components, you must ensure that any dependencies are first installed, as listed in Table 54.

   To install and use the package listed in the *Package Name* column, you must also install the components listed in the *Package Dependencies* column.

Table 54. Package component dependencies

Package Name	Component Function	Package Dependencies
ibmmq-runtime	Common function for all other components	None
ibmmq-client	C IBM MQ client libraries	ibmmq-runtime
ibmmq-java	Java and JMS IBM MQ APIs	ibmmq-runtime
ibmmq-jre	Java Runtime Environment	ibmmq-runtime
ibmmq-sdk	Header files and libraries for non-Java APIs	ibmmq-runtime
ibmmq-man	UNIX man pages for IBM MQ	ibmmq-runtime
ibmmq-samples	IBM MQ application samples	ibmmq-runtime
ibmmq-msg-cs ibmmq-msg-de ibmmq-msg-es ibmmq-msg-fr ibmmq-msg-hu ibmmq-msg-it ibmmq-msg-ja ibmmq-msg-ko ibmmq-msg-pl ibmmq-msg-pt ibmmq-msg-ru ibmmq-msg-zh-cn ibmmq-msg-zh-tw	Language specific message catalog files	ibmmq-runtime
ibmmq-gskit	IBM Global Security Kit	ibmmq-runtime ibmmq-jre

Table 54. Package component dependencies (continued)

Package Name	Component Function	Package Dependencies
V 9.0.2 ibmmq-sfbridge	IBM MQ Bridge to Salesforce **Note:** The IBM MQ Bridge to Salesforce is available only on Linux for System x (64 bit).	ibmmq-runtime ibmmq-java ibmmq-jre
V 9.0.4 ibmmq-bcbridge	IBM MQ Bridge to blockchain **Note:** The IBM MQ Bridge to blockchain is available only on Linux for System x (64 bit).	ibmmq-runtime ibmmq-java ibmmq-jre

Results

You have installed the packages you require.

What to do next

- If you have chosen this installation to be the primary installation on the system, you must now set it as the primary installation. Enter the following command at the command prompt:

  ```
  MQ_INSTALLATION_PATH/bin/setmqinst -i -p MQ_INSTALLATION_PATH
  ```
- You might want to set up the environment to work with this installation. You can use the **setmqenv** or **crtmqenv** command to set various environment variables for a particular installation of IBM MQ. For more information, see setmqenv and crtmqenv.
- For instructions on how to verify your installation, see “Testing communication between a client and a server on Linux” on page 393

Related concepts:

“Multiple installations on UNIX, Linux, and Windows” on page 200
On UNIX, Linux, and Windows, it is possible to have more than one copy of IBM MQ on a system.

“Primary installation on UNIX, Linux, and Windows” on page 202
On systems that support multiple installations of IBM MQ (UNIX, Linux, and Windows), the primary installation is the one to which IBM MQ system-wide locations refer. Having a primary installation is optional, but convenient.

Related tasks:

“Uninstalling or modifying IBM MQ on Linux using rpm” on page 395
On Linux, you can uninstall the IBM MQ server or client using the **rpm** command. You can also modify an installation by removing selected packages (components) currently installed on your system.

Related information:

setmqinst
Changing the primary installation

Redistributable clients on Linux

Linux

The Linux x86-64 image is shipped in a `LinuxX64.tar.gz` file.

File names

The archive or .zip file names describe the file contents and equivalent maintenance levels. For example, in IBM MQ Version 8.0.0, Fix Pack 4 the client images are available under the following file names:

Linux x86-64
`8.0.0.4-WS-MQC-Redist-LinuxX64.tar.gz`

Java - all platforms
`8.0.0.4-WS-MQC-Redist-Java.zip`

Choosing the runtime files to distribute with an application

A script file named **genmqpkg** is provided by the redistributable client under the `bin` directory.

You can use the **genmqpkg** script to generate a smaller subset of files that are tailored to the needs of the application, for which the files are intended to be distributed.

You are asked a series of interactive `Yes` or `No` questions to determine the runtime requirements for an IBM MQ application.

Finally, **genmqpkg** asks you to supply a new target directory, where the script duplicates the required directories and files.

Important: A fully qualified path should be supplied to **genmqpkg**, as **genmqpkg** will not expand or evaluate shell variables.

Important: IBM support is only able to provide assistance with the full, unmodified set of files contained within the redistributable client packages.

Other considerations

There is a minor change to the default data path of a non-installed client, and now the path is, on:

Linux x86-64
`$HOME/IBM/MQ/data`

A redistributable client runtime co-exists with a full IBM MQ client or server installation, provided that they are installed in different locations.

Important: Unpacking a redistributable image into the same location as a full IBM MQ installation is not supported.

On Linux the `ccsid.tbl` used to define the supported CCSID conversions is traditionally expected to be found in the `UserData` directory structure, along with error logs, trace files, and so on.

The `UserData` directory structure is populated by unpacking the redistributable client, and so, if the file is not found in its usual location, the redistributable client falls back to locate the file in the `/lib` subdirectory of the installation.

Classpath changes

The classpath used by **dspmqver**, **setmqenv**, and **crtmqenv** commands, add the `com.ibm.mq.allclient.jar` to the environment, immediately following the `com.ibm.mq.jar` and `com.ibm.mqjms.jar`.

An example of **dspmqver** output from the redistributable client on Linux:

```
Name:        IBM MQ
Version:     8.0.0.4
Level:       p800-804-L150909
BuildType:   IKAP - (Production)
Platform:    IBM MQ for Linux (x86-64 platform)
Mode:        64-bit
O/S:         Linux 2.6.32.59-0.7-default
InstName:    MQNI08000004
```

```
InstDesc:    IBM MQ V8.0.0.4 (Redistributable)
Primary:     No
InstPath:    /Development/johndoe/unzip/unpack
DataPath:    /u/johndoe/IBM/MQ/data
MaxCmdLevel: 802
```

Related concepts:

"Redistributable clients on Linux" on page 374
The Linux x86-64 image is shipped in a `LinuxX64.tar.gz` file.

Converting a trial license on Linux

Linux

Convert a trial license to a full license without reinstalling IBM MQ.

When the trial license expires, the "count-down" displayed by the **strmqm** command informs you the license has expired, and the command does not run.

Before you begin

1. IBM MQ is installed with a trial license.
2. You have access to the installation media of a fully licensed copy of IBM MQ.

About this task

Run the **setmqprd** command to convert a trial license to a full license.

If you do not want to apply a full license to your trial copy of IBM MQ, you can uninstall it at any time.

Procedure

1. Obtain the full license from the fully licensed installation media.

   The full license file is `amqpcert.lic`. On Linux, it is in the *`/MediaRoot/`*`licenses` directory on the installation media.
2. Run the **setmqprd** command from the installation that you are upgrading:
   ```
   MQ_INSTALLATION_PATH/bin/setmqprd /MediaRoot/licenses/amqpcert.lic
   ```

Related information:

setmqprd

Displaying messages in your national language on Linux

Linux

To display messages from a different national language message catalog, you must install the appropriate catalog and set the **LANG** environment variable.

About this task

Messages in U.S. English are automatically installed with IBM MQ

Message catalogs for all languages are installed in *`MQ_INSTALLATION_PATH`*`/msg/`*`language identifier`*, where *language identifier* is one of the identifiers in Table 55 on page 377.

If you require messages in a different language, perform the following steps:

Procedure

1. Install the appropriate message catalog (see "IBM MQ components and features" on page 192).

2. To select messages in a different language, ensure the **LANG** environment variable is set to the identifier for the language you want to install:

Table 55. Language identifiers

Identifier	Language
cs_CZ	Czech
de_DE	German
es_ES	Spanish
fr_FR	French
hu_HU	Hungarian
it_IT	Italian
ja_JP	Japanese
ko_KR	Korean
pl_PL	Polish
pt_BR	Brazilian Portuguese
ru_RU	Russian
zh_CN	Simplified Chinese
zh_TW	Traditional Chinese

Verifying an IBM MQ installation on Linux

Linux

The topics in this section provide instructions on how to verify a server or a client installation of IBM MQ on Linux systems.

About this task

You can verify a local (stand-alone) server installation or a server-to-server installation of the IBM MQ server:
- A local server installation has no communication links with other IBM MQ installations.
- A server-to-server installation does have links to other installations.

You can also verify that your IBM MQ MQI client installation completed successfully and that the communication link is working.

Procedure

- To verify a local server installation, see “Verifying a local server installation on Linux” on page 378.
- To verify a server-to-server installation, see “Verifying a server-to-server installation on Linux” on page 381.
- To verify a client installation, see “Verifying a client installation on Linux” on page 387.

Verifying a local server installation on Linux

You can use either the command line or the postcard application to verify a local (stand-alone) installation on Linux.

About this task

You can use the command line to verify that IBM MQ is successfully installed, and that the associated communication links are working properly.

You can also verify an installation using the postcard application. The postcard application is Java based and requires a system with the ability to view a graphical display.

Procedure

- To use the command line to verify an installation, see "Verifying a local server installation using the command line on Linux."
- To use the postcard application to verify an installation, see "Verifying a local server installation using the Postcard application on Linux" on page 380.

Verifying a local server installation using the command line on Linux:

On Linux systems, you can verify a local installation by using the command line to create a simple configuration of one queue manager and one queue. You can also verify an installation using the postcard application.

Before you begin

To verify the installation, you must first install the samples package.

Before beginning the verification procedure, you might want to check that you have the latest fixes for your system. For more information about where to find the latest updates, see "Checking requirements on Linux" on page 330.

About this task

Use the following steps to configure your default queue manager from the command line. After the queue manager is configured, use the `amqsput` sample program to put a message on the queue. You then use the `amqsget` sample program to get the message back from the queue.

IBM MQ object definitions are case-sensitive. Any text entered as an MQSC command in lowercase is converted automatically to uppercase unless you enclose it in single quotation marks. Make sure that you type the examples exactly as shown.

Procedure

1. On a Linux system, log in as a user in the `mqm` group.
2. Set up your environment:
   a. Set up environment variables for use with a particular installation by entering one the following command:
      ```
      . MQ_INSTALLATION_PATH/bin/setmqenv -s
      ```
      where *`MQ_INSTALLATION_PATH`* refers to the location where IBM MQ is installed.
   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 a queue manager called `QMA` by entering the following command:
   ```
   crtmqm QMA
   ```
   Messages indicate when the queue manager is created, and when the default IBM MQ objects are created.
4. Start the queue manager by entering the following command:
   ```
   strmqm QMA
   ```
   A message indicates when the queue manager starts.
5. Start MQSC by entering the following command:
   ```
   runmqsc QMA
   ```
   A message indicates when MQSC starts. MQSC has no command prompt.
6. Define a local queue called `QUEUE1` by entering the following command:
   ```
   DEFINE QLOCAL (QUEUE1)
   ```
   A message indicates when the queue is created.
7. Stop MQSC by entering the following command:
   ```
   end
   ```
   Messages are shown, followed by the command prompt.

Note: Subsequent steps require that the samples package is installed.

8. Change into the *`MQ_INSTALLATION_PATH`*`/samp/bin` directory, which contains the sample programs. *`MQ_INSTALLATION_PATH`* represents the high-level directory in which IBM MQ is installed.
9. Put a message on the queue by entering the following commands
   ```
   ./amqsput QUEUE1 QMA
   ```
   The following messages are shown:
   ```
   Sample AMQSPUT0 start
   target queue is QUEUE1
   ```
10. Type some message text on one or more lines, where each line is a different message. 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. Get the messages from the queue, by entering the following command:
    ```
    ./amqsget QUEUE1 QMA
    ```
    The sample program starts, and your messages are displayed.

Results

You have successfully verified your local installation.

Verifying a local server installation using the Postcard application on Linux: Linux

Sending messages successfully between two Postcard applications verifies a local installation.

Before you begin

The postcard application is Java based and requires a system with the ability to view a graphical display.

You must ensure that you are a member of the IBM MQ administrators group (**mqm**).

Note: Using Postcard to verify an IBM MQ installation is only possible if there is one IBM MQ installation on that box. The Default Configuration wizard will not create a default configuration if a queue manager already exists on the box. The Default Configuration wizard will run on any installation on a box but only one default configuration can be created per box. Using Postcard to verify second and subsequent installations of IBM MQ on the same box is not possible.

To verify that the local installation is working, you can run two instances of the Postcard application on the same server. The postcard application can send messages to, and receive messages from, other postcard applications. Successful sending and receiving of messages verifies that IBM MQ is installed and working correctly on the server.

Procedure

1. Log on as a user in group **mqm**.
2. Start the postcard application in one of the following ways:
   a. From the command line:
      1) Change the directory to *MQ_INSTALLATION_PATH*/java/bin. *MQ_INSTALLATION_PATH* represents the high-level directory in which IBM MQ is installed.
      2) Run the postcard application by entering the following command:
         ```
         ./postcard
         ```
   b. From the IBM MQ Explorer: On Linux (x86-64 platforms), you can start IBM MQ Explorer by using the system menu, the **MQExplorer** command (preferred command), or the MQExplorer executable file. The **strmqcfg** command is still usable.
      1) If the Welcome to IBM MQ Explorer Content view page does not show, click **IBM MQ** in the Navigator view to show the Welcome page.
      2) Click **Launch Postcard** to start the Postcard.
3. At the Postcard - Sign On window, type in a nickname to use to send messages within the Postcard application (for example, User1).
4. Select the queue manager to use as the mailbox:
   - If you do not have any queue managers, you are prompted to either launch the Default Configuration or close the Postcard application. Launching the Default Configuration creates a default queue manager.
   - If the only queue manager on your server is the default queue manager, this queue manager is used automatically for the postcard application. The default queue manager is created by running the Default Configuration wizard
   - If you have created your own queue managers, but you have not run the Default Configuration wizard, select an appropriate queue manager from the list.
   - If you have run the Default Configuration wizard and you want to use the default queue manager, but there are other queue managers on your server, select the **Advanced** check box. Then select **Use Default Configuration as mailbox**.

- If you have run the Default Configuration wizard and also created your own queue managers, and you do not want to use the default queue manager, select the **Advanced** check box. Then select **Choose queue manager as mailbox**, and then select the appropriate queue manager from the list.

   When your selection is complete, click **OK** to display your first Postcard window.

5. Run a second instance of the Postcard application by following the steps used to open the first instance of the Postcard application.
6. The Postcard - Sign On panel is displayed again. Type in a second nickname to use to send messages within this second Postcard application (for example, `User2`).
7. Repeat the selection of the queue manager that you want to use as the mailbox (as described in step 4). The queue manager you select for this second Postcard must be the same queue manager as used for the first instance of the Postcard application.
8. In the first Postcard, (User1), enter the nickname (`User2`) for the second Postcard application in the **To:** field. Because the sender and receiver are on the same server, you can leave the **On:** field blank.
9. Type a message in the **Message:** field and click **Send**.
10. The **Postcards sent and received** area of the Postcard shows details of the message. In the sending Postcard, the message is displayed as sent. In the receiving Postcard, the message is displayed as received.
11. In the receiving Postcard, (User2), double-click the message in the **Postcards sent and received** area to view it. When this message arrives, it verifies that IBM MQ is correctly installed.

What to do next

Depending on your situation, you might want to do the following tasks:
- Install IBM MQ on other servers. Follow the installation procedure for the appropriate platform. Ensure that you use the **Join Default Cluster** window in the Default Configuration wizard to add the other servers to the cluster on your first server.
- Install the IBM MQ MQI client on other servers.
- Continue with further administration tasks, see Administering IBM MQ.

Verifying a server-to-server installation on Linux

Linux

You can use either the command line or the postcard application to verify a server-to-server installation on Linux.

Before you begin

For a server-to-server verification, the communication links between the two systems must be checked. Before you can do the verification, you must therefore ensure that the communications protocol is installed and configured on both systems.

On Linux, IBM MQ supports TCP on all Linux platforms. On x86 platforms and Power platforms, SNA is also supported. If you want to use the SNA LU6.2 support on these platforms, you need the IBM Communications Server for Linux Version 6.2. The Communications Server is available as a PRPQ product from IBM. For more details, see Communications Server.

The examples in this task use TCP/IP. If you do not use TCP, see Setting up communication on UNIX and Linux.

About this task

For a server-to server installation, you can use the command line to verify that IBM MQ is successfully installed, and that the associated communication links are working properly.

You can also verify an installation using the postcard application. The postcard application is Java based and requires a system with the ability to view a graphical display.

Procedure

- To use the command line to verify an installation, see "Verifying a server-to-server installation using the command line on Linux."
- To use the postcard application to verify an installation, see "Verifying a server-to-server installation using the Postcard application on Linux" on page 385.

Verifying a server-to-server installation using the command line on Linux: Linux

You can verify a server-to-server installation using two servers, one as a sender and one as a receiver.

Before you begin

- Make sure that TCP/IP and IBM MQ are installed on both servers (see "Verifying a server-to-server installation on Linux" on page 381).
- Make sure that you are a member of the IBM MQ administrators group (**mqm**) on each server.
- Decide which installation is the sender server and which installation is the receiver server. The installations might be on the same system, or on different systems.

About this task

IBM MQ object definitions are case-sensitive. Any text entered as an MQSC command in lowercase is converted automatically to uppercase unless you enclose it in single quotation marks. Make sure that you type the examples exactly as shown.

Procedure

1. On the **receiver** server:
   a. On Linux, log in as a user in the mqm group.
   b. Check which ports are free, for example by running **netstat**. For more information about this command, see the documentation of your operating system.

      If port 1414 is not in use, make a note of 1414 to use as the port number in step 2 h. Use the same number for the port for your listener later in the verification. If it is in use, note a port that is not in use; for example 1415.
   c. Set up the environment for the installation you are using by entering the following command at the command prompt:
      ```
      . MQ_INSTALLATION_PATH/bin/setmqenv -s
      ```
      where *MQ_INSTALLATION_PATH* refers to the location where IBM MQ is installed.
   d. Create a queue manager called QMB by entering the following command at the command prompt:
      ```
      crtmqm QMB
      ```
      Messages tell you that the queue manager has been created, and that the default IBM MQ objects have been created.
   e. Start the queue manager by entering the following command:
      ```
      strmqm QMB
      ```

A message tells you when the queue manager has started.

f. Start MQSC by entering the following command:

```
runmqsc QMB
```

A message tells you that MQSC has started. MQSC has no command prompt.

g. Define a local queue called `RECEIVER.Q` by entering the following command:

```
DEFINE QLOCAL (RECEIVER.Q)
```

A message tells you the queue has been created.

h. Define a listener by entering the following command:

```
DEFINE LISTENER (LISTENER1) TRPTYPE (TCP) CONTROL (QMGR) PORT ( PORT_NUMBER )
```

Where *port_number* is the name of the port the listener runs on. This number must be the same as the number used when defining your sender channel.

i. Start the listener by entering the following command:

```
START LISTENER (LISTENER1)
```

Note: Do not start the listener in the background from any shell that automatically lowers the priority of background processes.

j. Define a receiver channel by entering the following command:

```
DEFINE CHANNEL (QMA.QMB) CHLTYPE (RCVR) TRPTYPE (TCP)
```

A message tells you when the channel has been created.

k. End MQSC by typing:

```
end
```

Some messages are displayed, followed by the command prompt.

2. On the **`sender`** server:

a. As the sender server is an AIX system, log in as a user in the `mqm` group.

b. Set up the environment for the installation you are using by entering the following command at the command prompt:

```
. MQ_INSTALLATION_PATH/bin/setmqenv -s
```

where *`MQ_INSTALLATION_PATH`* refers to the location where IBM MQ is installed.

c. Create a queue manager called `QMA` by entering the following command at the command prompt:

```
crtmqm QMA
```

Messages tell you that the queue manager has been created, and that the default IBM MQ objects have been created.

d. Start the queue manager, by entering the following command:

```
strmqm QMA
```

A message tells you when the queue manager has started.

e. Start MQSC by entering the following command:

```
runmqsc QMA
```

A message tells you that an MQSC session has started. MQSC had no command prompt.

f. Define a local queue called `QMB` (to be used as a transmission queue) by entering the following command:

```
DEFINE QLOCAL (QMB) USAGE (XMITQ)
```

A message tells you when the queue has been created.

g. Define a local definition of the remote queue with by entering the following command:

```
DEFINE QREMOTE (LOCAL.DEF.OF.REMOTE.QUEUE) RNAME (RECEIVER.Q) RQMNAME ('QMB') XMITQ (QMB)
```

h. Define a sender channel by entering one of the following commands:

con-name is the TCP/IP address of the receiver system. If both installations are on the same system, the *con-name* is `localhost`. *port* is the port you noted in 1 b. If you do not specify a port, the default value of 1414 is used.

```
DEFINE CHANNEL (QMA.QMB) CHLTYPE (SDR) CONNAME ('CON-NAME(PORT)') XMITQ (QMB) TRPTYPE (TCP)
```

i. Start the sender channel by entering the following command:

```
START CHANNEL(QMA.QMB)
```

The receiver channel on the receiver server starts automatically when the sender channel starts.

j. Stop MQSC by entering the following command:

```
end
```

Some messages are displayed, followed by the command prompt.

k. Change into the `MQ_INSTALLATION_PATH/samp/bin` directory. This directory contains the sample programs. `MQ_INSTALLATION_PATH` represents the high-level directory in which IBM MQ is installed.

l. If both the sender server and receiver server are installations on the same system, check that the queue managers have been created on different installations by entering the following command:

```
dspmq -o installation
```

If the queue managers are on the same installation, move either QMA to the sender installation or QMB to the receiver installation by using the **`setmqm`** command. For more information, see setmqm.

m. Put a message on the local definition of the remote queue, which in turn specifies the name of the remote queue. Enter the following command:

```
./amqsput LOCAL.DEF.OF.REMOTE.QUEUE QMA
```

A message tells you that `amqsput` has started.

n. Type some message text on one or more lines, followed by a blank line. A message tells you that `amqsput` has ended. Your message is now on the queue and the command prompt is displayed again.

3. On the **receiver** server:

a. As your receiver server is an AIX system, change into the `MQ_INSTALLATION_PATH/samp/bin` directory. This directory contains the sample programs. `MQ_INSTALLATION_PATH` represents the high-level directory in which IBM MQ is installed.

b. Get the message from the queue on the receiver by entering the following command:

```
./amqsget RECEIVER.Q QMB
```

The sample program starts, and your message is displayed. After a pause, the sample ends. Then the command prompt is displayed.

Results

You have now successfully verified the server-to-server installation.

Verifying a server-to-server installation using the Postcard application on Linux: Linux

You can use two instances of the Postcard application to verify that a server-to-server installation is working.

Before you begin

You can use the Postcard application on two servers, one instance of the Postcard application on each server, to verify that a server-to-server installation is working. Successful sending and receiving of messages verifies that IBM MQ is successfully installed, and that communication between the two servers is working correctly.

Note:
- If the system has multiple IBM MQ installations, ensure that Postcard has not been run before on any installations on that server. As the default configuration can only exist on one IBM MQ installation per system, the Default Configuration wizard and Postcard can not be used for verification of a second or any subsequent installation.
- The two server installations must be on different systems to do a server-to-server verification using the postcard application. To verify a server-to-server installation on the same machine, you can use the command line.
- Make sure that TCP/IP and IBM MQ are installed on both servers.
- Make sure that your systems are able to view a graphical display.
- Make sure that you are a member of the IBM MQ administrators group (**mqm**) on each server.
- Check that one of the following scenarios applies:
  - Neither server has had any queue managers created.
  - Use the Default Configuration wizard to create default queue managers on each server and link them to the default cluster.

    Details on how to use the Default Configuration wizard are provided in this topic.
  - Both servers have existing queue managers and these queue managers are in the same cluster.

    If your queue managers are not in the same cluster, create new queue managers on both servers. Then create a cluster, and ensure that the queue managers that you create on each server belong to that cluster.
  - You have configured channels to communicate between the two servers.

    For instructions on how to set up the channels, see “Verifying a server-to-server installation using the command line on Linux” on page 382. After you have set up the channels, follow the instructions in this topic to verify your server-to-server installation.

Procedure
1. On the first server, log on as a user in group **mqm**.
2. Start the postcard application in one of the following ways:
   a. From the command line:
      1) Change the directory to *MQ_INSTALLATION_PATH*/java/bin. *MQ_INSTALLATION_PATH* represents the high-level directory in which IBM MQ is installed.
      2) Run the postcard application by entering the following command:
      ```
      ./postcard
      ```
   b. From the IBM MQ Explorer: On Linux systems (x86-64 platforms), you can start IBM MQ Explorer by using the system menu, the MQExplorer executable file, or the **strmqcfg** command.
      1) If the Welcome to IBM MQ Explorer Content view page does not show, click **IBM MQ** in the Navigator view to show the Welcome page.
      2) Click **Launch Postcard** to start the Postcard.

3. At the Postcard - Sign On window, type a nickname to use to send messages within the Postcard application. For example, `User1` for the first server, and `User2` for the second server.
4. Select the queue manager to use as the mailbox:
   - If you do not have any queue managers, you are prompted to either launch the Default Configuration or close the Postcard application. Work through the Default Configuration wizard. When you get to the option to join the queue manager to the default cluster, tick the check box. On the next screen:
     - For the first server, select **yes, make it the repository for the cluster**.
     - For the second server, select **No another computer has already joined the cluster as a repository**. When requested, enter the location of the repository, by typing the name of the sender server.
   - If the only queue manager on your server is the default queue manager, this queue manager is used automatically for the postcard application. The default queue manager is created by running the Default Configuration wizard
   - If you have created your own queue managers, but you have not run the Default Configuration wizard, select an appropriate queue manager from the list.
   - If you have run the Default Configuration wizard and you want to use the default queue manager, but there are other queue managers on your server, select the **Advanced** check box. Then select **Use Default Configuration as mailbox**.
   - If you have run the Default Configuration wizard and also created your own queue managers, and you do not want to use the default queue manager, select the **Advanced** check box. Then select **Choose queue manager as mailbox**, and then select the appropriate queue manager from the list.

   When your selection is complete, click **OK**.
5. Select the queue manager to use as the mailbox:
   - If you do not have any queue managers, you are prompted to either launch the Default Configuration or close the Postcard application. Work through the Default Configuration wizard. When you get to the option to join the queue manager to the default cluster, tick the check box. On the next screen:
     - For the first server, select **yes, make it the repository for the cluster**.
     - For the second server, select **No another computer has already joined the cluster as a repository**. When requested, enter the location of the repository, by typing the name of the sender server.
   - If the only queue manager on your server is the default queue manager, this queue manager is used automatically for the postcard application. The default queue manager is created by running the Default Configuration wizard
   - If you have created your own queue managers, but you have not run the Default Configuration wizard, select an appropriate queue manager from the list.
   - If you have run the Default Configuration wizard and you want to use the default queue manager, but there are other queue managers on your server, select the **Advanced** check box. Then select **Use Default Configuration as mailbox**.
   - If you have run the Default Configuration wizard and also created your own queue managers, and you do not want to use the default queue manager, select the **Advanced** check box. Then select **Choose queue manager as mailbox**, and then select the appropriate queue manager from the list.

   When your selection is complete, click **OK**.
6. Complete steps 1 - 5 for the second server.
7. In the Postcard on the first server:
   a. Enter the nickname (`user2`) for the Postcard application on the second server in the **To:** field.
   b. Enter the queue manager on the second server in the **On:** field.
   c. Type a message in the **Message:** field and click **Send**.

8. In the Postcard on the second server:
   a. In the **Postcards sent and received**, double-click the message marked as received to view the message from the first server.
   b. Optional: Send a postcard to the first server by adapting the instructions in step 7. You must enter details of the first server in the **To:** field and the **On:** field.

   The messages verify that IBM MQ is correctly installed and that your communication link between the two servers is working correctly.

Verifying a client installation on Linux

▶ Linux

You can verify that your IBM MQ MQI client installation completed successfully and that the communication link is working.

About this task

The verification procedure shows how to create a queue manager called `queue.manager.1`, a local queue called `QUEUE1`, and a server-connection channel called `CHANNEL1` on the server.

It shows how to create the client-connection channel on the IBM MQ MQI client workstation. It then shows how to use the sample programs to put a message onto a queue, and get the message from the queue.

The example does not address any client security issues. See Setting up IBM MQ MQI client security for details if you are concerned with IBM MQ MQI client security issues.

The verification procedure assumes that:
- The full IBM MQ server product has been installed on a server.
- The server installation is accessible on your network.
- The IBM MQ MQI client software has been installed on a client system.
- The IBM MQ sample programs have been installed.
- TCP/IP has been configured on the server and client systems. For more information, see Configuring connections between the server and client.

Procedure

1. Set up the server and client:
   - To set up the server and client by using the command line, follow the instructions in "Setting up the server and client using the command line on Linux" on page 388.
   - To set up the server and client by using IBM MQ Explorer, follow the instructions in "Setting up the server and client using IBM MQ Explorer on Linux" on page 391.
2. Test the communications between client and server, using the instructions in "Testing communication between a client and a server on Linux" on page 393.

Related tasks:

"Installing an IBM MQ client on Linux" on page 355
Installing an IBM MQ client on a 32 bit or 64 bit Linux system.

Setting up the server and client using the command line on Linux: Linux

You can use the command line to create the objects that you need to use to verify a client installation on Linux. On the server you create a queue manager, a local queue, a listener, and a server-connection channel. You must also apply security rules to allow the client to connect and make use of the queue defined. On the client you create a client-connection channel. After setting up the server and client, you can then use the sample programs to complete the verification procedure.

Before you begin

Before starting this task, review the information in "Verifying a client installation on Linux" on page 387.

About this task

This task explains how to use the command line to set up the server and client so that you can verify your client installation.

If you prefer to use IBM MQ Explorer, see "Setting up the server and client using IBM MQ Explorer on Linux" on page 391.

Procedure

1. Set up the server by following the instructions in "Setting up the server using the command line on Linux."
2. Set up the client by following instructions in "Connecting to a queue manager, using the `MQSERVER` environment variable on Linux" on page 390.

What to do next

Test the communications between client and server by following the instructions in "Testing communication between a client and a server on Linux" on page 393.

Setting up the server using the command line on Linux: Linux

Follow these instructions to create a queue manager, queue, and channel on the server. You can then use these objects to verify the installation.

About this task

These instructions assume that no queue manager or other IBM MQ objects have been defined.

IBM MQ object definitions are case-sensitive. Any text entered as an MQSC command in lowercase is converted automatically to uppercase unless you enclose it in single quotation marks. Make sure that you type the examples exactly as shown.

Procedure

1. Create a user ID on the server that is not in the `mqm` group. This user ID must exist on the server and client. This is the user ID that the sample applications must be run as, otherwise a 2035 error is returned.
2. Log in as a user in the mqm group.

3. You must set various environment variables so that the installation can be used in the current shell. You can set the environment variables by entering the following command:

```
. MQ_INSTALLATION_PATH/bin/setmqenv -s
```

   where `MQ_INSTALLATION_PATH` refers to the location where IBM MQ is installed.
4. Create a queue manager called `QUEUE.MANAGER.1` by entering the following command:

```
crtmqm QUEUE.MANAGER.1
```

   You see messages telling you that the queue manager has been created.
5. Start the queue manager by entering the following command:

```
strmqm QUEUE.MANAGER.1
```

   A message tells you when the queue manager has started.
6. Start MQSC by entering the following command:

```
runmqsc QUEUE.MANAGER.1
```

   A message tells you that an MQSC session has started. MQSC has no command prompt.
7. Define a local queue called `QUEUE1` by entering the following command:

```
DEFINE QLOCAL(QUEUE1)
```

   A message tells you when the queue has been created.
8. Allow the user ID that you created in step 1 to use `QUEUE1` by entering the following command:

```
SET AUTHREC PROFILE(QUEUE1) OBJTYPE(QUEUE) PRINCIPAL(' non_mqm_user ') AUTHADD(PUT,GET)
```

   where *non_mqm_user* is the user ID created in step 1. A message tells you when the authorization has been set. You must also run the following command to give the user ID authority to connect:

```
SET AUTHREC OBJTYPE(QMGR) PRINCIPAL(' non_mqm_user ') AUTHADD(CONNECT)
```

   If this command is not run, a 2305 stop error is returned.
9. Define a server-connection channel by entering the following command:

```
DEFINE CHANNEL (CHANNEL1) CHLTYPE (SVRCONN) TRPTYPE (TCP)
```

   A message tells you when the channel has been created.
10. Allow your client channel to connect to the queue manager and run under the user ID that you created in step 1, by entering the following MQSC command:

```
SET CHLAUTH(CHANNEL1) TYPE(ADDRESSMAP) ADDRESS(' client_ipaddr ') MCAUSER(' non_mqm_user ')
```

   where *client_ipaddr* is the IP address of the client system, and *non_mqm_user* is the user ID created in step 1. A message tells you when the rule has been set.
11. Define a listener by entering the following command:

```
DEFINE LISTENER (LISTENER1) TRPTYPE (TCP) CONTROL (QMGR) PORT (port_number)
```

   where *port_number* is the number of the port the listener is to run on. This number must be the same as the number used when defining your client-connection channel in "Installing an IBM MQ client on Linux" on page 355.

   Note: If you omit the port parameter from the command, a default value of `1414` is used for the listener port. If you want to specify a port other than `1414`, you must include the port parameter in the command, as shown.
12. Start the listener by entering the following command:

```
START LISTENER (LISTENER1)
```

13. Stop MQSC by entering:

```
end
```

You see some messages, followed by the command prompt.

What to do next

Follow the instructions to set up the client. See "Connecting to a queue manager, using the `MQSERVER` environment variable on Linux."

Connecting to a queue manager, using the `MQSERVER` environment variable on Linux: Linux

When an IBM MQ application is run on the IBM MQ MQI client, it requires the name of the MQI channel, the communication type, and the address of the server to be used. Provide these parameters by defining the `MQSERVER` environment variable.

Before you begin

Before you start this task, you must complete the task, "Setting up the server using the command line on Linux" on page 388, and save the following information:
- The host name or IP address of the server and port number that you specified when creating the listener.
- The channel name of the server-connection channel.

About this task

This task describes how to connect an IBM MQ MQI client, by defining the `MQSERVER` environment variable on the client.

You can give the client access to the generated client channel definition table, `amqclchl.tab` instead; see Accessing client-connection channel definitions.

Procedure
1. Log in as the userid that you created in Step 1 of "Setting up the server using the command line on Linux" on page 388.
2. Check the TCP/IP connection. From the client, enter one of the following commands:
   - `ping server-hostname`
   - `ping n.n.n.n`

     `n.n.n.n` represents the network address. You can set the network address in IPv4 dotted decimal form, for example, `192.0.2.0`. Alternatively, set the address in IPv6 hexadecimal form, for example `2001:0DB8:0204:acff:fe97:2c34:fde0:3485`.

   If the **`ping`** command fails, correct your TCP/IP configuration.
3. Set the `MQSERVER` environment variable. From the client, enter the following command:
   ```
   export MQSERVER=CHANNEL1/TCP/'server-address (port)'
   ```

   Where:
   - *CHANNEL1* is the server-connection channel name.
   - *server-address* is the TCP/IP host name of the server.
   - *port* is the TCP/IP port number the server is listening on.

   If you do not give a port number, IBM MQ uses the one specified in the `qm.ini` file, or the client configuration file. If no value is specified in these files, IBM MQ uses the port number identified in the TCP/IP services file for the service name `MQSeries`. If an `MQSeries` entry in the services file does

not exist, a default value of 1414 is used. It is important that the port number used by the client and the port number used by the server listener program are the same.

What to do next

Use the sample programs to test communication between the client and server; see "Testing communication between a client and a server on Linux" on page 393.

Setting up the server and client using IBM MQ Explorer on Linux: Linux

You can use IBM MQ Explorer to create the objects that you need to use to verify a client installation on Linux. On the server, you create a queue manager, a local queue, a listener and a server-connection channel. On the client system you create a client-connection channel. Then from the command line you use the sample PUT and GET programs to complete the verification procedure.

Before you begin

Before starting this task, review the information in "Verifying a client installation on Linux" on page 387.

About this task

This task explains how to use IBM MQ Explorer to set up the server and client so that you can verify your client installation.

If you prefer to use the command line, see "Setting up the server and client using the command line on Linux" on page 388.

Procedure
1. Set up the server by following the instructions in "Setting up the server using IBM MQ Explorer on Linux."
2. Set up the client by following instructions in "Setting up the client using IBM MQ Explorer on Linux" on page 392.

What to do next

Test the communications between client and server by following the instructions in "Testing communication between a client and a server on Linux" on page 393.

Related tasks:

"Installing an IBM MQ client on Linux" on page 355

Installing an IBM MQ client on a 32 bit or 64 bit Linux system.

Setting up the server using IBM MQ Explorer on Linux: Linux

You can use the IBM MQ Explorer to create the server objects that you need to verify your client installation.

About this task

To verify your installation, you must first create a queue manager, a local queue, a listener and a server-connection channel on the server.

Procedure
1. Create a queue manager:
   a. Open IBM MQ Explorer.

   b. Right-click the folder called **Queue Managers**, select **New** > **Queue Manager**.
   c. In the first entry field, type the queue manager name, *QUEUE.MANAGER.1*, and click **Finish**.
2. Create a local queue:
   a. Expand the queue manager you have just created and right-click **queues**.
   b. Select **New** > **Local Queue**.
   c. Enter the queue name, *QUEUE1*, and click **Finish**.
3. Define the server-connection channel:
   a. Right-click **Channels**.
   b. Select **New** > **Server Connection Channel**.
   c. Enter the channel name, *CHANNEL1*, and click **Next**.
   d. In the dialog navigation pane, click **MCA** to open the MCA page.
   e. In the MCA User ID field, enter a userid that is a member of the mqm group, typically your own.
   f. Click **Finish**.
4. Run the listener.

   The listener is automatically started when the queue manager is configured. To check that the listener is running, open **Listeners** and look for `LISTENER.TCP`.

What to do next

Set up the client. See "Setting up the client using IBM MQ Explorer on Linux."

Related tasks:

"Installing an IBM MQ client on Linux" on page 355
Installing an IBM MQ client on a 32 bit or 64 bit Linux system.

Setting up the client using IBM MQ Explorer on Linux: Linux

You can use IBM MQ Explorer to define the client-connection if you are setting up the client and server on the same workstation on a Linux system.

Procedure

1. Select the queue manager, *QUEUE.MANAGER.1*
2. Open the **Channels** folder, then right-click **Client Connections** > **New** > **Client-connection Channel...**
3. Enter the channel name, *CHANNEL1*, for the client connection, and click **Next**.
4. Enter the queue manager name, *QUEUE.MANAGER.1*
5. Enter the following string as the connection name:
   ```
    server-address (port)
   ```
   Where:
   - `server-address` is the TCP/IP host name of the server
   - `port` is the TCP/IP port number the server is listening on
6. Click Finish.
7. From the command line, set the `MQCHLLIB` environment variable: Enter the following command:
   ```
   export MQCHLLIB=var/mqm/qmgrs/QUEUE!MANAGER!1/@ipcc
   ```
   Note: The queue manager name contains "**.**". IBM MQ creates the queue manager directory with the name, `QUEUE!MANAGER!1`

What to do next

Use the sample programs to test communication between the client and server. See "Testing communication between a client and a server on Linux."

Related tasks:

"Installing an IBM MQ client on Linux" on page 355

Installing an IBM MQ client on a 32 bit or 64 bit Linux system.

Testing communication between a client and a server on Linux: Linux

On the IBM MQ MQI client workstation, use the amqsputc sample program to put a message on the queue at the server workstation. Use the amqsgetc sample program to get the message from the queue back to the client.

Before you begin

Complete the previous topics in this section:
- Set up a queue manager, channels, and queue.
- Open a command window.
- Set system environment variables.

About this task

Note that IBM MQ object definitions are case-sensitive. Text entered as an MQSC command in lowercase is converted automatically to uppercase unless you enclose it in single quotation marks. Make sure that you type the examples exactly as shown.

You must be logged in with the appropriate authority. For example, user `ivtid` in the `mqm` group.

Procedure

1. Change to the `*MQ_INSTALLATION_PATH*/samp/bin directory`, which contains the sample programs. *`MQ_INSTALLATION_PATH`* represents the high-level directory in which IBM MQ is installed.
2. You must set certain environment variables so that the installation can be used in the current shell. You can set the environment variables by entering the following command:
   ```
   . MQ_INSTALLATION_PATH/bin/setmqenv -s
   ```
   where *`MQ_INSTALLATION_PATH`* refers to the location where IBM MQ is installed.
3. Start the PUT program for `QUEUE1` on `QUEUE.MANAGER.1` by entering the following command:
   ```
   ./amqsputc QUEUE1 QUEUE.MANAGER.1
   ```
   If the command is successful, the following messages are displayed:
   ```
   Sample AMQSPUT0 start
   target queue is QUEUE1
   ```
   Tip: You might get the error, `MQRC_NOT_AUTHORIZED` (2035). By default, channel authentication is enabled when a queue manager is created. Channel authentication prevents privileged users accessing a queue manager as an IBM MQ MQI client. For verifying the installation, you can either change the MCA user ID to a non-privileged user, or disable channel authentication. To disable channel authentication run the following MQSC command:
   ```
   ALTER QMGR CHLAUTH(DISABLED)
   ```
   When you finish the test, if you do not delete the queue manager, re-enable channel authentication:
   ```
   ALTER QMGR CHLAUTH(ENABLED)
   ```

4. Type some message text, then press **Enter** twice. The following message is displayed:
   ```
   Sample AMQSPUT0 end
   ```
   Your message is now on the queue that is on the server queue manager.
5. Start the GET program for QUEUE1 on QUEUE.MANAGER.1 by entering the following command:
   ```
   ./amqsgetc QUEUE1 QUEUE.MANAGER.1
   ```
   The sample program starts, and your message is displayed. After a short pause (approximately 30 seconds), the sample ends and the command prompt is displayed again.

Results

You have now successfully verified the client installation.

What to do next

1. You must set various environment variables on the server so that the installation can be used in the current shell. You can set the environment variables by entering the following command:
   ```
   . MQ_INSTALLATION_PATH/bin/setmqenv -s
   ```
   where *MQ_INSTALLATION_PATH* refers to the location where IBM MQ is installed.
2. On the server, stop the queue manager by entering the following command:
   ```
   endmqm QUEUE.MANAGER.1
   ```
3. On the server, delete the queue manager by entering the following command:
   ```
   dltmqm QUEUE.MANAGER.1
   ```

Uninstalling or modifying IBM MQ on Linux

▶ Linux

You can uninstall an IBM MQ server or client. You can also modify an installation by removing selected packages (components) currently installed on your system.

Procedure

For information on how to uninstall or modify IBM MQ on Linux, see the following subtopics:
- "Uninstalling or modifying IBM MQ on Linux using rpm" on page 395
- "Uninstalling or modifying IBM MQ on Linux Ubuntu using Debian packages" on page 396

Uninstalling or modifying IBM MQ on Linux using rpm

Linux

On Linux, you can uninstall the IBM MQ server or client using the **rpm** command. You can also modify an installation by removing selected packages (components) currently installed on your system.

Before you begin

If you have applied one or more fix packs to the version of IBM MQ that you want to uninstall, you need to remove the fix packs in reverse chronological installation order before you remove the base packages.

You must remove any updates before starting the uninstallation procedure. For more information, see Restoring the previous maintenance level on IBM MQ on Linux.

Important: You must stop all IBM MQ queue managers, other objects, and applications, before you begin the process to uninstall or modify IBM MQ.

Procedure

1. Stop all IBM MQ applications associated with the installation you are uninstalling or modifying, if you have not already done so.
2. For a server installation, end any IBM MQ activity associated with the installation you are uninstalling or modifying:
   a. Log in as a user in the group `mqm`.
   b. Set up your environment to work with the installation you want to uninstall or modify. Enter the following command:
      ```
      . MQ_INSTALLATION_PATH/bin/setmqenv -s
      ```
      where *MQ_INSTALLATION_PATH* refers to the location where IBM MQ is installed.
   c. Display the state of all queue managers on the system. Enter the following command:
      ```
      dspmq -o installation
      ```
   d. Stop all running queue managers associated with the installation you want to uninstall or modify. Enter the following command for each queue manager:
      ```
      endmqm QMgrName
      ```
   e. Stop any listeners associated with the queue managers. Enter the following command for each queue manager:
      ```
      endmqlsr -m QMgrName
      ```
3. Log in as root.
4. Uninstall or modify IBM MQ using the **rpm** command:
   a. On a system with a single installation:
      - Find out the names of the packages (components) currently installed on your system, by entering the following command:
        ```
        rpm -qa | grep MQSeries
        ```
      - Remove all components by appending all the package names to the **rpm** command arguments. For example:
        ```
        rpm -qa | grep MQSeries | xargs rpm -ev
        ```
      - Modify your installation by appending individual package names to the rpm command arguments. For example, to remove the runtime, Server and SDK components enter the following command:
        ```
        rpm -ev MQSeriesRuntime MQSeriesServer MQSeriesSDK
        ```
      - If you are using Ubuntu, add the **--force-debian** attribute. For example, to remove the runtime, Server and SDK components enter the following command:

```
rpm --force-debian -ev MQSeriesRuntime MQSeriesServer MQSeriesSDK
```

b. On a system with multiple installations:
   - Find out the names of the packages (components) currently installed on your system, by entering the following command:

     ```
     rpm -qa | grep suffix
     ```

     where *suffix* is the unique name given to the packages when **crtmqpkg** was run at installation time. *suffix* is included in each of the package names that belong to a particular installation.
   - Remove all components by appending all the package names to the **rpm** command arguments. For example, to remove all components from an installation with the suffix MQ80 enter the following command:

     ```
     rpm -qa | grep '\<MQSeries.*MQ80\>' | xargs rpm -ev
     ```
   - Modify your installation by appending individual package names to the **rpm** command arguments. For example, to remove the runtime, Server and SDK components from an installation with the suffix MQ80 enter the following command:

     ```
     rpm -ev MQSeriesRuntime-MQ80 MQSeriesServer-MQ80 MQSeriesSDK-MQ80
     ```
   - If you are using Ubuntu, add the **--force-debian** attribute. For example, to remove the runtime, Server and SDK components for an installation with the *suffix* MQ80, enter the following command:

     ```
     rpm --force-debian -ev MQSeriesRuntime-MQ80 MQSeriesServer-MQ80 MQSeriesSDK-MQ80
     ```

Results

After uninstallation, certain files under the directory trees `/var/mqm` and `/etc/opt/mqm` are not removed. These files contain user data and remain so subsequent installations can reuse the data. Most of the remaining files contain text, such as INI files, error logs, and FDC files. The directory tree `/var/mqm/shared` contains files that are shared across installations, including the executable shared libraries `libmqzsd.so` and `libmqzsd_r.so`.

What to do next

- If the product successfully uninstalled, you can delete any files and directories contained in the installation directory.
- If there are no other IBM MQ installations on the system, and you are not planning to reinstall or migrate, you can delete the `/var/mqm` and `/etc/opt/mqm` directory trees, including the files `libmqzsd.so` and `libmqzsd_r.so`. Deleting these directories destroys all queue managers and their associated data.

Uninstalling or modifying IBM MQ on Linux Ubuntu using Debian packages

Linux

You can uninstall an IBM MQ server or client that was installed using the Debian package manager. You can also modify an installation by removing selected packages (components) currently installed on your system.

Before you begin

If you have applied one or more fix packs to the version of IBM MQ that you want to uninstall, you need to remove the fix packs in reverse chronological installation order before you remove the base packages.

You must remove any updates before starting the uninstallation procedure. For more information, see Restoring the previous maintenance level on IBM MQ on Linux.

Important: You must stop all IBM MQ queue managers, other objects, and applications, before you begin the process to uninstall or modify IBM MQ.

Procedure

1. Stop all IBM MQ applications associated with the installation you are uninstalling or modifying, if you have not already done so.
2. For a server installation, end any IBM MQ activity associated with the installation you are uninstalling or modifying:
   a. Log in as a user in the group `mqm`.
   b. Set up your environment to work with the installation you want to uninstall or modify. Enter the following command:
   ```
   . MQ_INSTALLATION_PATH/bin/setmqenv -s
   ```
   where *MQ_INSTALLATION_PATH* refers to the location where IBM MQ is installed.
   c. Display the state of all queue managers on the system. Enter the following command:
   ```
   dspmq -o installation
   ```
   d. Stop all running queue managers associated with the installation you want to uninstall or modify. Enter the following command for each queue manager:
   ```
   endmqm QMgrName
   ```
   e. Stop any listeners associated with the queue managers. Enter the following command for each queue manager:
   ```
   endmqlsr -m QMgrName
   ```
3. Log in as root.
4. Uninstall or modify IBM MQ using a Debian installation command:
   - Using **dpkg**.
     Issuing the command:
     ```
     dpkg -r packagename
     ```
     removes the product but leaves the package definition cached.
     Issuing the command:
     ```
     dpkg -P packagename
     ```
     purges the cached definition of the product.
   - Using **apt**.
     Issuing the command:
     ```
     apt remove "ibmmq-*"
     ```
     removes the product but leaves the package definition cached.
     Issuing the command:
     ```
     apt purge "ibmmq-*"
     ```
     purges the cached definition of the product.

Results

After uninstallation, certain files under the directory trees `/var/mqm` and `/etc/opt/mqm` are not removed. These files contain user data and remain so subsequent installations can reuse the data. Most of the remaining files contain text, such as INI files, error logs, and FDC files. The directory tree `/var/mqm/shared` contains files that are shared across installations, including the executable shared libraries `libmqzsd.so` and `libmqzsd_r.so`.

What to do next

- If the product successfully uninstalled, you can delete any files and directories contained in the installation directory.

- If there are no other IBM MQ installations on the system, and you are not planning to reinstall or migrate, you can delete the `/var/mqm` and `/etc/opt/mqm` directory trees, including the files `libmqzsd.so` and `libmqzsd_r.so`. Deleting these directories destroys all queue managers and their associated data.

Installing and uninstalling IBM MQ on Solaris

Solaris

Installation tasks that are associated with installing IBM MQ on Solaris systems are grouped in this section.

About this task

To prepare for installation and to install the IBM MQ components, complete the following tasks.

For information about how to uninstall IBM MQ, see “Uninstalling IBM MQ on Solaris” on page 434.

If product fixes or updates are made available, see IBM MQ maintenance tasks for information about how to apply these changes.

Procedure

1. Check the system requirements. See “Checking requirements on Solaris” on page 403.
2. Plan your installation.
   - As part of the planning process, you must choose which components to install and where to install them. See “IBM MQ components for Solaris systems.”
   - You must also make some platform-specific choices. See “Planning to install IBM MQ on Solaris” on page 404.
3. Prepare your system for installation of IBM MQ. See “Preparing the system on Solaris” on page 405.
4. Install IBM MQ server. See “Installing IBM MQ server on Solaris” on page 410.
5. Optional: Install an IBM MQ client. See “Installing an IBM MQ client on Solaris” on page 416.
6. Verify your installation. See “Verifying an IBM MQ installation on Solaris” on page 420.

IBM MQ components for Solaris systems

Solaris

You can select the components that you require when you install IBM MQ.

Important: See IBM MQ license information for details of what each purchase of IBM MQ entitles you to install.

Table 56 on page 399 shows the components that are available when installing an IBM MQ server or client on a Solaris system.

Note: When you install interactively on Solaris systems, the options that are available install various combinations of the components listed in this table. Details are given in the “Interactive installation” on page 400 section.

Table 56. IBM MQ components for Solaris systems

Component	Description	Server media	Client media	Component name
Runtime	Contains files that are common to both server and client installations. **Note:** This component must be installed.	✓	✓	runtime
Server	You can use the server to run queue managers on your system and connect to other systems over a network. Provides messaging and queuing services to applications, and support for IBM MQ client connections.	✓		server
Standard Client	The IBM MQ MQI client is a small subset of IBM MQ, without a queue manager, that uses the queue manager and queues on other (server) systems. It can be used only when the system it is on is connected to another system that is running a full server version of IBM MQ. The client and the server can be on the same system if required.	✓	✓	sol_client
SDK	The SDK is required for compiling applications. It includes sample source files, and the bindings (files .H, .LIB, .DLL, and others), that you need to develop applications to run on IBM MQ.	✓	✓	base
Sample programs	The sample application programs are needed if you want to check your IBM MQ installation using the verification procedures.	✓	✓	samples
Java messaging	The files needed for messaging using Java (includes Java Message Service).	✓	✓	java
Man pages	UNIX man pages, in U.S. English, for: control commands MQI calls MQSC commands	✓	✓	man
Java JRE	A Java Runtime Environment that is used by those parts of IBM MQ that are written in Java.	✓	✓	jre
Message Catalogs	For available languages, see the table of message catalogs that follows.	✓	✓	
IBM Global Security Kit	IBM Global Security Kit V8 Certificate and TLS, Base Runtime.	✓	✓	gskit
Managed File Transfer	MQ Managed File Transfer transfers files between systems in a managed and auditable way, regardless of file size or the operating systems used. For information about the function of each component, see Managed File Transfer product options.	✓		ftagent ftbase ftlogger ftservice fttools

Table 56. IBM MQ components for Solaris systems (continued)

Component	Description	Server media	Client media	Component name
Advanced Message Security	Provides a high level of protection for sensitive data flowing through the IBM MQ network, while not impacting the end applications. You must install this component on all IBM MQ installations that host queues you want to protect. You must install the IBM Global Security Kit component on any IBM MQ installation that is used by a program that puts or gets messages to or from a protected queue, unless you are using only Java client connections. You must install the **Java JRE** component to install this component.	✓		mqams
V 9.0.0 AMQP Service	Install this component to make AMQP channels available. AMQP channels support MQ Light APIs. You can use AMQP channels to give AMQP applications access to the enterprise-level messaging facilities provided by IBM MQ.	✓		amqp

Table 57. IBM MQ message catalogs for Solaris systems.

A two-column table listing the available message catalogs.

Message catalog language	Component name
Brazilian Portuguese	Pt_BR
Czech	Cs_CZ
French	Fr_FR
German	De_DE
Hungarian	Hu_HU
Italian	It_IT
Japanese	Ja_JP
Korean	Ko_KR
Polish	Pl_PL
Russian	Ru_RU
Spanish	Es_ES
Simplified Chinese	Zh_CN
Traditional Chinese	Zh_TW
U.S. English	not applicable

Interactive installation

The options available with interactive installation install various combinations of the product components described in the previous tables. The following table shows you what will be installed for each option, together with the option number on the server and client DVDs:

Table 58. IBM MQ interactive installation options for Solaris systems.

A four-column table listing interactive installation options and the components installed with each one. Server and client option numbers are also listed.

Interactive installation option	**Components installed**
IBM MQ Server	base runtime server java gskit
Man pages	runtime man
Sample programs	base runtime samples
IBM MQ MQI client libraries (including Java, JMS, and Web Services support)	base runtime sol_client java gskit
IBM Java runtime for Solaris, Java 2 Technology Edition, Version 6	jre runtime
IBM Global Security Kit for IBM MQ	gskit jre runtime
Managed File Transfer Service	ftservice ftbase jre java runtime ftagent
Managed File Transfer Tools	fttools ftbase jre java runtime
Managed File Transfer Agent	ftagent ftbase jre java runtime
Managed File Transfer Logger	ftlogger ftbase jre java runtime server
Advanced Message Security	runtime mqams
V 9.0.0 AMQP Service	runtime jre java amqp

Table 58. IBM MQ interactive installation options for Solaris systems (continued).

A four-column table listing interactive installation options and the components installed with each one. Server and client option numbers are also listed.

Interactive installation option	**Components installed**
V 9.0.1 REST API and Console	runtime jre java web
Spanish message catalog	runtime Es_ES
French message catalog	runtime Fr_FR
German message catalog	runtime De_DE
Japanese message catalog	runtime Ja_JP
Italian message catalog	runtime It_IT
Brazilian Portuguese message catalog	runtime Pt_BR
Traditional Chinese message catalog	runtime Zh_TW
Simplified Chinese message catalog	runtime Zh_CN
Korean message catalog	runtime Ko_KR
Russian message catalog	runtime Ru_RU
Hungarian message catalog	runtime Hu_HU
Polish message catalog	runtime Pl_PL
Czech message catalog	runtime Cs_CZ

Related concepts:

"IBM MQ components and features" on page 192
You can select the components or features that you require when you install IBM MQ.

"Planning considerations for installation on Multiplatforms" on page 196
Before you install IBM MQ, you must choose which components to install and where to install them. You must also make some platform-specific choices.

Checking requirements on Solaris

Solaris

Before you install IBM MQ on Solaris, you must check for the latest information and system requirements.

About this task

A summary of the tasks that you must complete to check system requirements are listed here with links to further information.

Procedure

1. Check that you have the latest information, including information on hardware and software requirements. See "Where to find product requirements and support information" on page 195.
2. Check that your systems meet the initial hardware and software requirements for Solaris. See "Hardware and software requirements on Solaris systems."

   The supported hardware and software environments are occasionally updated. See System Requirements for IBM MQ for the latest information.
3. Check that your systems have sufficient disk space for the installation. See Disk space requirements.
4. Check that you have the correct licenses. See "License requirements" on page 194 and IBM MQ license information.

What to do next

When you have completed these tasks, you are ready to start preparing your system for installation. For the next steps in installing IBM MQ, see "Preparing the system on Solaris" on page 405.

Related concepts:

"IBM MQ installation overview" on page 192
An overview of concepts and considerations for installing IBM MQ, with links to instructions on how to install, verify, and uninstall IBM MQ on each of the supported platforms.

Related information:

IBM MQ maintenance tasks

Hardware and software requirements on Solaris systems

Solaris

Before you install IBM MQ , check that your system meets the hardware and operating system software requirements for the particular components you intend to install.

For hardware and software requirements, see System Requirements for IBM MQ.

IBM MQ does not support host names that contain spaces. If you install IBM MQ on a system with a host name that contains spaces, you are unable to create any queue managers.

Java Message Service and SOAP transport

If you want to use Java Message Service and SOAP support, you need an IBM Java 7 SDK and Runtime Environment Version 7.0 or later.

V 9.0.0 Java 8 is bundled with IBM MQ Version 9.0 but client components are built with Java 7 compatibility flags on.

For development, a JDK is required, and a JRE is required for running. The JRE does not need to be the JRE installed with IBM MQ, but has to be one from the supported list.

For a list of supported JDKs, see System Requirements for IBM MQ.

For further information about SOAP with IBM MQ , see IBM MQ transport for SOAP.

On Solaris : The 32-bit and 64-bit JDKs are typically installed to the same directory. To run a 64-bit JVM use the -d64 or -d32 parameters on the command line when running a Java application to ensure the correct JVM is used.

You can check the version installed using the following command:

```
java -version
```

Transport Layer Security (TLS)

If you want to use the TLS support, you need the IBM Global Security Kit (GSKit) V8 package. This package is supplied with IBM MQ as one of the components available for installation.

Solaris 11 operating system

If you are installing on the Solaris 11 operating system, ensure that the IPS package (package/svr4) that supports `pkgadd` and equivalent utilities is installed.

Related concepts:

IBM i “Hardware and software requirements on IBM i systems” on page 299
Check that the server environment meets the prerequisites for installing IBM MQ for IBM i. Check the product readme files and install missing prerequisite software supplied on the server CD.

“Hardware and software requirements on Windows systems” on page 445
Check that the server environment meets the prerequisites for installing IBM MQ for Windows and install any prerequisite software that is missing from your system from the server DVD.

Related tasks:

“Checking requirements on Windows” on page 444
Before you install IBM MQ on Windows, you must check for the latest information and system requirements.

Planning to install IBM MQ on Solaris

Solaris

Before you install IBM MQ on Solaris, you must choose which components to install and where to install them. You must also make some platform-specific choices.

About this task

The following steps provide links to additional information to help you with planning your installation of IBM MQ on Solaris.

As part of your planning activities, make sure that you review the information on hardware and software requirements for the platform on which you are planning to install IBM MQ. For more information, see "Checking requirements on Solaris" on page 403.

Procedure

1. Decide which IBM MQ components and features to install. See "IBM MQ components and features" on page 192.

   Important: Ensure that your enterprise has the correct license, or licenses, for the components that you are going to install. For more information, see "License requirements" on page 194 and IBM MQ license information.
2. Review the options for naming your installation. In some cases, you can choose an installation name to use instead of the default name. See "Installation name on UNIX, Linux, and Windows" on page 197.
3. Review the options and restrictions for choosing an installation location for IBM MQ. For more information, see "Installation location on Multiplatforms" on page 198.
4. If you plan to install multiple copies of IBM MQ, see "Multiple installations on UNIX, Linux, and Windows" on page 200.
5. If you already have a primary installation, or plan to have one, see "Primary installation on UNIX, Linux, and Windows" on page 202.
6. Make sure that the communications protocol needed for server-to-server verification is installed and configured on both systems that you plan to use. For more information, see "Server-to-server links on UNIX, Linux, and Windows" on page 210.

Preparing the system on Solaris

Solaris

On Solaris systems, you might have to complete several tasks before you install IBM MQ. You might also want to complete other tasks, depending on your installation intentions.

About this task

The tasks that you perform to prepare your systems for installation are listed here. Complete the appropriate tasks for your platform before installing.

Procedure

1. Set up a user ID of the name `mqm`, with a primary group of `mqm`. See "Setting up the user and group on Solaris" on page 406.
2. Create file systems for both the product code and working data to be stored. See "Creating file systems on Linux" on page 335.
3. Configure any additional settings needed for your Solaris system. See "Configuring and tuning the operating system on Solaris" on page 409.

What to do next

When you have completed the tasks to prepare the system, you are ready to start installing IBM MQ. To install a server, see "Installing IBM MQ server on Solaris" on page 410. To install a client, see "Installing an IBM MQ client on Solaris" on page 416.

Related information:
Planning
Maintaining and migrating
IBM MQ maintenance tasks

Setting up the user and group on Solaris

Solaris

On Solaris systems, IBM MQ requires a user ID of the name `mqm`, with a primary group of `mqm`. The `mqm` user ID owns the directories and files that contain the resources associated with the product.

Creating the user ID and group

Set the primary group of the `mqm` user to the group `mqm`.

If you are installing IBM MQ on multiple systems you might want to ensure each UID and GID of `mqm` has the same value on all systems. If you are planning to configure multi-instance queue managers, it is essential the UID and GID are the same from system to system. It is also important to have the same UID and GID values in virtualization scenarios.

Solaris

The user ID value for user `mqm` must be less than 262,143 to avoid problems with the maintenance update process.

Create the IDs using the **`groupadd`** and **`useradd`** commands to set the UID and GID the same on each machine.

Adding existing user IDs to the group on Solaris systems

If you want to run administration commands, for example **`crtmqm`** (create queue manager) or **`strmqm`** (start queue manager), your user ID must be a member of the `mqm` group. This user ID must not be longer than 12 characters.

Users do not need `mqm` group authority to run applications that use the queue manager; it is needed only for the administration commands.

Log files created by MQ Telemetry service

The **`umask`** setting of the user ID that creates a queue manager will determine the permissions of the Telemetry log files generated for that queue manager. Even though the ownership of the log files will be set to `mqm`.

Related concepts:

"Creating file systems on AIX" on page 223
Before installing IBM MQ, you might need to create file systems for both the product code and working data to be stored. There are minimum storage requirements for these file systems. The default installation directory for the product code can be changed at installation time, but the working data location cannot be changed.

"Configuring and tuning the operating system on HP-UX" on page 271
Before you install IBM MQ on an HP-UX system, you must check that the kernel is configured correctly.

"Configuring and tuning the operating system on Linux" on page 337
Use this topic when you are configuring IBM MQ on Linux systems.

Related tasks:

"Configuring and tuning the operating system on AIX" on page 224
When installing IBM MQ on AIX systems, there are some additional settings that must be configured.

Related information:

"Configuring and tuning the operating system on Solaris" on page 409
Configure Solaris systems with the resource limits required by IBM MQ.

Creating file systems on Solaris

▶ Solaris

Before installing IBM MQ, you might need to create file systems for both the product code and working data to be stored. There are minimum storage requirements for these file systems. The default installation directory for the product code can be changed at installation time, but the working data location cannot be changed.

Determining the size of a server installations file system

To determine the size of the `/var/mqm` file system for a server installation, consider:

- The maximum number of messages in the system at one time.
- Contingency for message buildups, if there is a system problem.
- The average size of the message data, plus 500 bytes for the message header.
- The number of queues.
- The size of log files and error messages.
- The amount of trace that is written to the `/var/mqm/trace` directory.

Storage requirements for IBM MQ also depend on which components you install, and how much working space you need. For more details, see Disk space requirements.

Creating a file system for the working data

Before you install IBM MQ, create and mount a file system called `/var/mqm` which is owned by the user `mqm` in the group `mqm` ; see "Setting up the user and group on Linux" on page 334. This file system is used by all installations of IBM MQ on a system. If possible, use a partition strategy with a separate volume for the IBM MQ data. This means that other system activity is not affected if a large amount of IBM MQ work builds up. Configure the directory permissions to permit the `mqm` user to have full control, for example, file mode 755. These permissions will then be updated during the IBM MQ installation to match the permissions required by the queue manager.

Creating separate file systems for errors and logs

You can also create separate file systems for your log data (`/var/mqm/log`) and error files (`/var/mqm/errors`). If possible, place these directories on different physical disks from the queue manager data (`/var/mqm/qmgrs`) and from each other.

If you create separate file systems the /var/mqm/errors directory can be NFS mounted. However, if you choose to NFS-mount /var/mqm/errors, the error logs might be lost if the network fails.

You can protect the stability of your queue manager by having separate file systems for:
- /var/mqm/errors
- /var/mqm/trace
- /var/mqm/qmgrs
- /var/mqm/log

In the case of /var/mqm/errors, it is rare that this directory receives large quantities of data. But it is sometimes seen, particularly if there is a severe system problem leading to IBM MQ writing a lot of diagnostic information in to .FDC files. In the case of /var/mqm/trace, files are only written here when you use **strmqtrc** to start tracing IBM MQ.

You can obtain better performance of normal IBM MQ operations (for example, syncpoints, MQPUT, MQGET of persistent messages) by placing the following on separate disks:
- /var/mqm/qmgrs
- /var/mqm/log

In the rare event that you need to trace an IBM MQ system for problem determination, you can reduce performance impact by placing the /var/mqm/trace file system on a separate disk.

If you are creating separate file systems, allow a minimum of 30 MB of storage for /var/mqm, 100 MB of storage for /var/mqm/log, and 10 MB of storage for /var/mqm/errors. The 100 MB minimum allowance of storage for /var/mqm/log is the absolute minimum required for a single queue manager and is not a recommended value. The size of a file system must be scaled according to the number of queue managers that you intend to use, the number of pages per log file, and the number of log files per queue manager.

If you want to use individual queues that hold more than 2 GB of data, you must enable /var/mqm to use large files.

For more information about file systems, see File system support.

The size of the log file depends on the log settings that you use. The minimum sizes are for circular logging using the default settings. For more information about log sizes, see Calculating the size of the log.

Solaris

For a client installation, the file system can be mounted on a remote network device, for example NFS.

If you are performing both a client and a server installation, the requirements of the server installation take precedence over the requirements of the client installation.

Allow 15 MB as a minimum for an IBM MQ client.

A new sample IBM MQ MQI client configuration file is created in the var/mqm directory, by the client package, during installation, but only if this file does not exist. This file contains the ClientExitPath stanza. An example mqclient.ini file is shown in Configuring a client using a configuration file.

If you are using a common configuration file for multiple clients, either in the IBM MQ installation directory or in another location using the MQCLNTCF environment variable, you must grant read access to all user identifiers under which the IBM MQ client applications run. If, for any reason, the file cannot be read the failure is traced, and the search logic continues as if the file had not existed.

Related concepts:

"Setting up the user and group on Solaris" on page 406
On Solaris systems, IBM MQ requires a user ID of the name `mqm`, with a primary group of `mqm`. The `mqm` user ID owns the directories and files that contain the resources associated with the product.

Related information:

"Configuring and tuning the operating system on Solaris"
Configure Solaris systems with the resource limits required by IBM MQ.

Configuring and tuning the operating system on Solaris

Solaris

Configure Solaris systems with the resource limits required by IBM MQ.

IBM MQ uses semaphores, shared memory, and file descriptors, and it is probable that the default resource limits are not adequate.

For further information on **maxusers**, and other process-sizing parameters, see Process sizing parameters.

To set new default limits for all users in the *mqm* group, set up a project for the *mqm* group in each zone.

To find out if you already have a project for the *mqm* group, log in as root and enter the following command:

```
projects -l
```

If you do not already have a *group.mqm* project defined, enter the following command:

```
projadd -c "IBM MQ default settings"
        -K "process.max-file-descriptor=(basic,10000,deny)"
        -K "project.max-shm-memory=(priv,4GB,deny)"
        -K "project.max-shm-ids=(priv,1024,deny)"
        -K "project.max-sem-ids=(priv,128,deny)" group.mqm
```

If a project called *group.mqm* is listed, review the attributes for that project. The attributes must include the following minimum values:

```
process.max-file-descriptor=(basic,10000,deny)
project.max-sem-ids=(priv,128,deny)
project.max-shm-ids=(priv,1024,deny)
project.max-shm-memory=(priv,4294967296,deny)
```

If you need to change any of these values, enter the following command:

```
projmod -s -K "process.max-file-descriptor=(basic,10000,deny)"
           -K "project.max-shm-memory=(priv,4GB,deny)"
           -K "project.max-shm-ids=(priv,1024,deny)"
           -K "project.max-sem-ids=(priv,128,deny)" group.mqm
```

Note that you can omit any attributes from this command that are already correct.

For example, to change only the number of file descriptors, enter the following command:

```
projmod -s -K "process.max-file-descriptor=(basic,10000,deny)" group.mqm
```

(To set only the limits for starting the queue manager under the mqm user, login as mqm and enter the command `projects`. The first listed project is likely to be `default`, and so you can use `default` instead of `group.mqm`, with the `projmod` command.)

To ensure that the attributes for the project `group.mqm` are used by a user session when running IBM MQ, make sure that the primary group of that user ID is mqm. In the examples in this topic, the `group.mqm` project ID will be used.

For further information on how projects are associated with user sessions, see System Administration Guide: Oracle Solaris Containers-Resource Management and Oracle Solaris Zones for your release of Solaris.

You can check your system configuration using the mqconfig command.

For more information on configuring your system, see How to configure UNIX and Linux systems for IBM MQ.

Related concepts:

"Setting up the user and group on Solaris" on page 406
On Solaris systems, IBM MQ requires a user ID of the name `mqm`, with a primary group of `mqm`. The `mqm` user ID owns the directories and files that contain the resources associated with the product.

"Creating file systems on AIX" on page 223
Before installing IBM MQ, you might need to create file systems for both the product code and working data to be stored. There are minimum storage requirements for these file systems. The default installation directory for the product code can be changed at installation time, but the working data location cannot be changed.

Installing IBM MQ server on Solaris

Solaris

You can install an IBM MQ server on Solaris either interactively or silently.

Before you begin

- Before you start the installation procedure, make sure that you complete the necessary steps that are outlined in "Preparing the system on Solaris" on page 405.
- If you install a copy of IBM MQ server for Solaris by using Electronic Software Download, obtained from Passport Advantage, you need to decompress the `tar.gz` file, and extract the installation files from the tar file, by using the following command:

  ```
  tar -xvf   WS_MQ_V8.0_TRIAL_FOR_SOLARIS_ML.tar
  ```

  Important: You must use GNU tar (also known as `gtar`) to unpack the `tar` images.
- If you are using Solaris zones, you have a choice between installing IBM MQ into the global zone, or installing IBM MQ into a non-global zone.

  For more information on how to install IBM MQ into Solaris zones, see the following technote: WebSphere MQ support position regarding Solaris zones. The technote is applicable to IBM WebSphere MQ Version 7.1 or later with the following changes:
  - You do not need the `-G` option on the **`pkgadd`** command as GSKit is now installed as part of the IBM MQ installation.
  - If you install IBM MQ into the global zone for use in sparse zones, you must copy the `/var/mqm` file system into the sparse zone. You must also copy the `/etc/opt/mqm/mqinst.ini` installation entry into the sparse zone.
  - Limitations for shared `/usr` file systems: the **`dspmqinst`** and **`dspmqver`** commands might report the primary installation incorrectly when compared with the symbolic links in `/usr/bin`. To synchronize the reporting of the primary installation in a Solaris zone and the global zone, run **`setmqinst`** with the **`-i`** or **`-x`** parameter, on the individual zones.
  - You cannot change the primary installation within a non-global zone. You must change the primary installation through the global zone, which has the appropriate write access to `/usr/bin`.

About this task

This task describes the installation of the IBM MQ for Solaris server, by using the `pkgadd` program. You can choose which components you want to install. The components are listed in “IBM MQ components for Solaris systems” on page 398.

Note: If you are installing on the Solaris 11 operating system, ensure that the IPS package (package/svr4) that supports `pkgadd` and equivalent utilities is installed.

Procedure

1. Log in as root, or switch to the superuser by using the **su** command.
2. Set your current directory to the location of the installation file. The location might be the mount point of the server DVD, a network location, or a local file system directory.
3. Run the `mqlicense.sh` script to accept the license:

   ```
   ./mqlicense.sh
   ```

   If you want to view a text-only version of the license, which can be read by a screen reader, type:

   ```
   ./mqlicense.sh -text_only
   ```

   The license is displayed. Follow the instructions to accept the license. If you accept the license, the installation continues. If you do not accept the license, you cannot continue the installation process.
4. If this installation is not the first installation on the system, run **crtmqpkg** to create a unique set of packages to install on the system:
   a. Enter the following command:

      ```
      ./crtmqpkg suffix
      ```

      where *suffix* is a name of your choosing that uniquely identifies the installation packages on the system. *suffix* is not the same as an installation name, although the names can be identical. *suffix* is limited to 16 characters in the ranges A-Z, a-z, and 0-9.

      The **crtmqpkg** script can use two environment variables that are useful when you are installing from a non-disk media location:
      - *CDROOT*, the root of the installation media or downloaded installation files.
      - *TMPDIR*, the output location of the modified installation files.

      No environment variables are required if you are running the image as `./crtmqpkg`.
   b. Set your current directory to the location specified when the **crtmqpkg** command completes. This directory is a subdirectory of `/var/spool`, in which the unique set of packages is created. The packages have the *suffix* value contained within the file name.
5. Start the installation process:
   - If the installation is the first installation on the system, enter the following command to start the installation process:

     ```
     pkgadd -d.
     ```

     where " . " means use the current directory.
   - If the installation is not the first installation on the system, enter the following command to start the installation process:

     ```
     pkgadd mqm-suffix
     ```

     where *suffix* is the suffix that is chosen in the previous step.
6. When prompted, choose a location for installation.
   - To install to the default location, `/opt/mqm`, enter `y`.

- To install to a non-default directory, enter `n` then enter the required installation path, and confirm your choice.

7. When the list of components is displayed, enter the numbers of the components that you require, separated by spaces or commas. If you are installing (adding) an IBM MQ component to an existing installation, choose option `yes` when you are asked whether to overwrite.

   Note: During the IBM MQ base version installation, you can choose to install all components or a subset of the components. When you install a fix pack, only the currently installed components are upgraded. If, at a later stage, you want to add further IBM MQ components that are not already installed, these components can be installed (added) to the IBM MQ base version only. If your current version of IBM MQ is not the base version, you must first uninstall all the fix packs before you add the required components to the existing installation, and then install the required fix packs. Also, when you are adding IBM MQ components to an existing installation, you must choose option `yes` when you are asked whether to overwrite by the installation process.
8. If the path chosen in step 6 does not exist, and you are asked if you want to create it, enter `y` to proceed.
9. Answer any questions appropriately for your system. If you are prompted to choose whether to install certain IBM MQ files as `setuid/setgid` files, you must enter `y`.
10. When a message informing you that the installation is complete appears, enter `q` to exit the `pkgadd` program.

What to do next

- If you have chosen this installation to be the primary installation on the system, you must now set it as the primary installation by entering the following command at the command line:
  ```
  MQ_INSTALLATION_PATH/bin/setmqinst -i -p MQ_INSTALLATION_PATH
  ```
  where *`MQ_INSTALLATION_PATH`* represents the directory where IBM MQ is installed.

  You can have only one primary installation on a system. If there is already a primary installation on the system, you must unset it before you can set another installation as the primary installation. For more information, see Changing the primary installation.
- You might want to set up the environment to work with this installation. You can use the **`setmqenv`** or **`crtmqenv`** command to set various environment variables for a particular installation of IBM MQ. For more information, see setmqenv and crtmqenv.
- If you want to confirm that the installation was successful, you can verify your installation. For more information, see "Verifying an IBM MQ installation on Solaris" on page 420.

Related concepts:

"Multiple installations on UNIX, Linux, and Windows" on page 200
On UNIX, Linux, and Windows, it is possible to have more than one copy of IBM MQ on a system.

"Primary installation on UNIX, Linux, and Windows" on page 202
On systems that support multiple installations of IBM MQ (UNIX, Linux, and Windows), the primary installation is the one to which IBM MQ system-wide locations refer. Having a primary installation is optional, but convenient.

Related tasks:

"Installing the server silently on Solaris" on page 413
You can perform a non-interactive installation of the IBM MQ server using the installation script `silent.sh`. A non-interactive installation is also known as a silent, or unattended installation.

"Uninstalling IBM MQ on Solaris" on page 434
On Solaris, you can uninstall the IBM MQ server or client using the **`pkgrm`** command.

Related information:

setmqinst

Changing the primary installation

Installing the server silently on Solaris

Solaris

You can perform a non-interactive installation of the IBM MQ server using the installation script `silent.sh`. A non-interactive installation is also known as a silent, or unattended installation.

Before you begin

Before you start the installation procedure, make sure that you have completed the necessary steps outlined in "Preparing the system on Solaris" on page 405.

About this task

You can perform a silent installation of IBM MQ. A sample script file called `silent.sh` is supplied in the `silent` directory on the DVD. You can use this script to perform a non-interactive installation that requires no input and shows nothing on the screen. It must be run as root.

The installation script `silent.sh` uses an `admin` file and a `response` file, both of which are supplied in the `silent` directory. You can use these files as supplied to perform a silent installation of all the components, including all the national language features, to the default location.

Note: If you are installing on the Solaris 11 operating system, ensure that the IPS package (package/svr4) that supports `pkgadd` and equivalent utilities is installed.

Procedure

1. Copy the `silent.sh` script into a writeable directory.
2. If this installation is not the first installation on the system, run **`crtmqpkg`** to create a unique set of packages to install on the system:
   a. Enter the following command:
   ```
   ./crtmqpkg suffix
   ```
   where *suffix* is a name of your choosing, that will uniquely identify the installation packages on the system. *suffix* is not the same as an installation name, although the names can be identical. *suffix* is limited to 16 characters in the ranges A-Z, a-z, and 0-9.
   b. Set your current directory to the location specified when the **`crtmqpkg`** command completes. This directory is a sub-directory of `/var/spool`, in which the unique set of packages is created. The packages have the *suffix* value contained within the filename.

   Once a new package has been generated for the second installation the `silent.sh` script needs to have its MQ_PACKAGE_NAME variable modified so that its value is not `mqm` but the new package name.

   Also the MQ_PACKAGE_LOCATION variable needs to be modified so that its value is not $MQ_MEDIA_LOCATION but the location of the new package (which by default is `/var/spool/pkg`).
3. Optional: If you want to change where the IBM MQ server DVD is mounted, update the values in the `silent.sh` script. By default, the script assumes that the server DVD has been mounted at `/CD7FVML`.
4. Optional: If you want to change where the output and logs are written to, update the values in the `silent.sh` script. By default, output and logs are written to the file `/var/tmp/mq.install`.
5. Optional: If you want to install to a non-default location, update the *`MQ_INSTALLATION_PATH`* variable in the `silent.sh` script.

   Note:
   - The installation path specified must either be an empty directory, the root of an unused file system, or a path that does not exist. The length of the path is limited to 256 bytes and must not contain spaces.

- If the directory you specified does not exist, the installation script creates that directory.

6. Optional: If you want to change the components that are installed, edit the `response` file. A list of all the installable IBM MQ components can be found at: "IBM MQ components and features" on page 192.

   Solaris does not check, during a silent installation, that prerequisite components are installed. You can use the following procedure to create a response file interactively, before using it to install the product. **`pkgask`** prompts you for the names of the components to install.

   a. Run the **`mqlicense.sh`** command to accept the license agreement for the product.
   b. **`pkgask`** -d *path_to_install_image* -r *response_file* `mqm`

   The inputs to **`pkgask`** are the same as those inputs documented for **`pkgadd`**, but instead of the product being installed a response file is created.
7. Optional: If you have edited the `response` file, you must then edit the `silent.sh` to use your custom response file.
8. To start the installation, run `silent.sh`.
9. Check the log file for any errors.

What to do next

- If you have chosen this installation to be the primary installation on the system, you must now set it as the primary installation by entering the following command at the command line:

  ```
  MQ_INSTALLATION_PATH/bin/setmqinst -i -p MQ_INSTALLATION_PATH
  ```

  where *`MQ_INSTALLATION_PATH`* represents the directory where IBM MQ is installed.

  You can have only one primary installation on a system. If there is already a primary installation on the system, you must unset it before you can set another installation as the primary installation. For more information, see Changing the primary installation.
- You might want to set up the environment to work with this installation. You can use the **`setmqenv`** or **`crtmqenv`** command to set various environment variables for a particular installation of IBM MQ . For more information, see setmqenv and crtmqenv.
- If you want to confirm that the installation was successful, you can verify your installation. See "Verifying an IBM MQ installation on Solaris" on page 420, for more information.

Related concepts:

"Multiple installations on UNIX, Linux, and Windows" on page 200
On UNIX, Linux, and Windows, it is possible to have more than one copy of IBM MQ on a system.

"Primary installation on UNIX, Linux, and Windows" on page 202
On systems that support multiple installations of IBM MQ (UNIX, Linux, and Windows), the primary installation is the one to which IBM MQ system-wide locations refer. Having a primary installation is optional, but convenient.

Related tasks:

"Installing IBM MQ server on Solaris" on page 410
You can install an IBM MQ server on Solaris either interactively or silently.

"Uninstalling IBM MQ on Solaris" on page 434
On Solaris, you can uninstall the IBM MQ server or client using the **`pkgrm`** command.

Related information:

setmqinst

Changing the primary installation

Converting a trial license on Solaris

▶ Solaris

Convert a trial license to a full license without reinstalling IBM MQ.

When the trial license expires, the "count-down" displayed by the **strmqm** command informs you the license has expired, and the command does not run.

Before you begin

1. IBM MQ is installed with a trial license.
2. You have access to the installation media of a fully licensed copy of IBM MQ.

About this task

Run the **setmqprd** command to convert a trial license to a full license.

If you do not want to apply a full license to your trial copy of IBM MQ, you can uninstall it at any time.

Procedure

1. Obtain the full license from the fully licensed installation media.

   The full license file is `amqpcert.lic`. On Solaris, it is in the */MediaRoot/*`licenses` directory on the installation media.
2. Run the **setmqprd** command from the installation that you are upgrading:

   ```
   MQ_INSTALLATION_PATH/bin/setmqprd /MediaRoot/licenses/amqpcert.lic
   ```

Related information:
setmqprd

Displaying messages in your national language on Solaris systems

Solaris

To display messages from a different national language message catalog, you must install the appropriate catalog and set the **LANG** environment variable.

About this task

Messages in U.S. English are automatically installed with IBM MQ

Message catalogs for all languages are installed in *MQ_INSTALLATION_PATH*`/msg/`*language identifier*, where *language identifier* is one of the identifiers in Table 59.

If you require messages in a different language, perform the following steps:

Procedure

1. Install the appropriate message catalog (see "IBM MQ components and features" on page 192).
2. To select messages in a different language, ensure the **LANG** environment variable is set to the identifier for the language you want to install:

Table 59. Language identifiers

Identifier	Language
cs_CZ	Czech
de_DE	German
es_ES	Spanish
fr_FR	French
hu_HU	Hungarian
it_IT	Italian

Table 59. Language identifiers (continued)

Identifier	Language
ja_JP	Japanese
ko_KR	Korean
pl_PL	Polish
pt_BR	Brazilian Portuguese
ru_RU	Russian
zh_CN	Simplified Chinese
zh_TW	Traditional Chinese

Installing an IBM MQ client on Solaris

Solaris

You can interactively install the IBM MQ client for Solaris using `pkgadd`.

Before you begin

- Before you start the installation procedure, make sure that you have completed the necessary steps outlined in “Preparing the system on Solaris” on page 405.
- This procedure is for the installation of a standard IBM MQ client, from the client DVD. If you are installing an IBM MQ client on a system that is already running an IBM MQ server, are therefore using a server DVD to install the client, follow the steps in “Installing IBM MQ server on Solaris” on page 410, and select the appropriate client components in step 8.

About this task

This task describes the installation of the IBM MQ for Solaris client, using the **pkgadd** program. You can choose which components you want to install. The components (or file sets) are listed in “IBM MQ components for Solaris systems” on page 398; you must install at least the Client component.

Note: If you are installing on the Solaris 11 operating system, ensure that the IPS package (package/svr4) that supports `pkgadd` and equivalent utilities is installed.

Procedure

1. Log in as root, or switch to the superuser using the **su** command.
2. Make your current directory the location of the installation file. The location might be the mount point of the DVD, a network location, or a local file system directory.
3. Run the `mqlicense.sh` script to accept the license:
   ```
   ./mqlicense.sh
   ```
   If you want to view a text-only version of the license, which can be read by a screen-reader, type:
   ```
   ./mqlicense.sh -text_only
   ```
   The license is displayed. Follow the instructions to accept the license. If you accept the license, the installation continues. If you do not accept the license, you cannot continue the installation process.
4. If this installation is not the first installation on the system, you must run **crtmqpkg** to create a unique set of packages to install on the system:
   a. Enter the following command:
      ```
      ./crtmqpkg suffix
      ```

where *suffix* is a name of your choosing, that will uniquely identify the installation packages on the system. *suffix* is not the same as an installation name, although the names can be identical. *suffix* is limited to 16 characters in the ranges A-Z, a-z, and 0-9.

   b. Set your current directory to the location specified when the **crtmqpkg** command completes. This directory is a sub-directory of `/var/spool`, in which the unique set of packages is created. The packages have the *suffix* value contained within the filename.

5. Start the installation process:
   - If the installation is the first installation on the system, enter the following command to start the installation process:
     ```
     pkgadd -d.
     ```
     where " **.** " means use the current directory.
   - If the installation is not the first installation on the system, enter the following command to start the installation process:
     ```
     pkgadd mqm-suffix
     ```
     where *suffix* is the suffix chosen in the previous step.
6. You are presented with a list of the packages that are available. Enter the number of the `mqm` package.
7. You are prompted to choose a location for installation.
   - To install to the default location, enter `y`.
   - To install to a non-default directory, enter `n`. Then enter the required installation path, and confirm your choice.
8. You receive a number of messages, after which a list of components is displayed. Enter the numbers of the components that you require separated by spaces or commas.
9. If the path chosen in step 7 does not exist, you are asked if you want to create it. You must enter `y` to proceed.
10. Answer any questions appropriately for your system.
11. A message tells you when installation is complete. Enter `q` to exit the `pkgadd` program.

What to do next

- If you have chosen this installation to be the primary installation on the system, you must now set it as the primary installation. Enter the following command at the command prompt:
  ```
  MQ_INSTALLATION_PATH/bin/setmqinst -i -p MQ_INSTALLATION_PATH
  ```
  You can have only one primary installation on a system. If there is already a primary installation on the system, you must unset it before you can set another installation as the primary installation. For more information, see Changing the primary installation.
- You might want to set up the environment to work with this installation. You can use the **setmqenv** or **crtmqenv** command to set various environment variables for a particular installation of IBM MQ. For more information, see setmqenv and crtmqenv.
- For instructions on how to verify your installation, see “Testing communication between a client and a server on Solaris” on page 432.

Related tasks:
"Uninstalling IBM MQ on Solaris" on page 434
On Solaris, you can uninstall the IBM MQ server or client using the **pkgrm** command.

Installing a client silently on Solaris

Solaris

You can perform a non-interactive installation of the IBM MQ client using the installation script `silent.sh`. A non-interactive installation is also known as a silent, or unattended installation.

Before you begin

- Before you start the installation procedure, make sure that you have completed the necessary steps outlined in "Preparing the system on Solaris" on page 405.
- This procedure is for the installation of a standard IBM MQ client, from the location of the installation file. The location might be the mount point of the DVD, a network location, or a local file system directory.
- This procedure is for the installation of a standard IBM MQ client, from the client DVD. If you are installing an IBM MQ client on a system that is already running an IBM MQ server, are therefore using a server DVD to install the client, follow the steps in "Installing IBM MQ server on Solaris" on page 410, and select the appropriate client components in step 8.

About this task

You can perform a silent installation of IBM MQ. A sample script file called `silent.sh` is supplied in the `silent` directory on the DVD. You can use this script to perform a non-interactive installation that requires no input and shows nothing on the screen. It must be run as root.

The installation script `silent.sh` uses an `admin` file and a `response` file, both of which are supplied in the `silent` directory. You can use these files as supplied to perform a silent installation of all the components, including all the national language features, to the default location.

Note: If you are installing on the Solaris 11 operating system, ensure that the IPS package (package/svr4) that supports `pkgadd` and equivalent utilities is installed.

Procedure

1. Copy the `silent.sh` script into a writeable directory.
2. If this installation is not the first installation on the system, run **crtmqpkg** to create a unique set of packages to install on the system:
   a. Enter the following command:
      ```
      ./crtmqpkg suffix
      ```
      where *suffix* is a name of your choosing, that will uniquely identify the installation packages on the system. *suffix* is not the same as an installation name, although the names can be identical. *suffix* is limited to 16 characters in the ranges A-Z, a-z, and 0-9.
   b. Set your current directory to the location specified when the **crtmqpkg** command completes. This directory is a sub-directory of `/var/spool`, in which the unique set of packages is created. The packages have the *suffix* value contained within the filename.

   Once a new package has been generated for the second installation the `silent.sh` script needs to have its MQ_PACKAGE_NAME variable modified so that its value is not `mqm` but the new package name.

   Also the MQ_PACKAGE_LOCATION variable needs to be modified so that its value is not $MQ_MEDIA_LOCATION but the location of the new package (which by default is `/var/spool/pkg`).

3. Optional: If you want to change where the IBM MQ client DVD is mounted, you must update the values in the `silent.sh` script. By default, the script assumes that the DVD has been mounted at `/CD7FVML`.
4. Optional: If you want to change where the output and logs are written to, update the values in the `silent.sh` script. By default, output and logs are written to the file `/var/tmp/mq.install`.
5. Optional: If you want to install to a non-default location, update the *`MQ_INSTALLATION_PATH`* variable in the `silent.sh` script.

   Note:
   - The installation path specified must either be an empty directory, the root of an unused file system, or a path that does not exist. The length of the path is limited to 256 bytes and must not contain spaces.
   - If the directory you specified does not exist, the installation script creates that directory.
6. Optional: If you want to change the components that are installed, edit the `response` file. A list of all the installable IBM MQ components can be found at: "IBM MQ components and features" on page 192.

   Solaris does not check, during a silent installation, that prerequisite components are installed. You can use the following procedure to create a response file interactively, before using it to install the product. **`pkgask`** prompts you for the names of the components to install.
   a. Run the **`mqlicense.sh`** command to accept the license agreement for the product.
   b. **`pkgask`** -d *path_to_install_image* -r *response_file* `mqm`

   The inputs to **`pkgask`** are the same as those inputs documented for **`pkgadd`**, but instead of the product being installed a response file is created.
7. Optional: If you have edited the `response` file, you must then edit the `silent.sh` to use your custom response file.
8. To start the installation, run `silent.sh`.
9. Check the log file for any errors.

What to do next

- If you have chosen this installation to be the primary installation on the system, you must now set it as the primary installation. Enter the following command at the command prompt:
  ```
  MQ_INSTALLATION_PATH/bin/setmqinst -i -p MQ_INSTALLATION_PATH
  ```
  You can have only one primary installation on a system. If there is already a primary installation on the system, you must unset it before you can set another installation as the primary installation. For more information, see Changing the primary installation.
- You might want to set up the environment to work with this installation. You can use the **`setmqenv`** or **`crtmqenv`** command to set various environment variables for a particular installation of IBM MQ. For more information, see setmqenv and crtmqenv.
- For instructions on how to verify your installation, see "Testing communication between a client and a server on Solaris" on page 432.

Verifying an IBM MQ installation on Solaris

Solaris

The topics in this section provide instructions on how to verify a server or a client installation of IBM MQ on Solaris systems.

About this task

You can verify a local (stand-alone) server installation or a server-to-server installation of the IBM MQ server:
- A local server installation has no communication links with other IBM MQ installations.
- A server-to-server installation does have links to other installations.

You can also verify that your IBM MQ MQI client installation completed successfully and that the communication link is working.

Procedure

- To verify a local server installation, see "Verifying a local server installation on Solaris."
- To verify a server-to-server installation, see "Verifying a server-to-server installation on Solaris" on page 424.
- To verify a client installation, see "Verifying a client installation using the command line on Solaris" on page 429.

Verifying a local server installation on Solaris

Solaris

You can use either the command line or the postcard application to verify a local (stand-alone) installation on Solaris.

About this task

You can use the command line to verify that IBM MQ is successfully installed, and that the associated communication links are working properly.

You can also verify an installation using the postcard application. The postcard application is Java based and requires a system with the ability to view a graphical display.

Procedure

- To use the command line to verify an installation, see "Verifying a local server installation using the command line on Solaris" on page 421.
- To use the postcard application to verify an installation, see "Verifying a local server installation using the Postcard application on Solaris" on page 422.

Verifying a local server installation using the command line on Solaris: ▶ Solaris

On Solaris systems, you can verify a local installation by using the command line to create a simple configuration of one queue manager and one queue. You can also verify an installation using the postcard application.

Before you begin

To verify the installation, you must first install the samples package.

Before beginning the verification procedure, you might want to check that you have the latest fixes for your system. For more information about where to find the latest updates, see “Checking requirements on Windows” on page 444.

About this task

Use the following steps to configure your default queue manager from the command line. After the queue manager is configured, use the `amqsput` sample program to put a message on the queue. You then use the `amqsget` sample program to get the message back from the queue.

IBM MQ object definitions are case-sensitive. Any text entered as an MQSC command in lowercase is converted automatically to uppercase unless you enclose it in single quotation marks. Make sure that you type the examples exactly as shown.

Procedure

1. If you are verifying an installation on a Solaris system, log in as a user in the `mqm` group.
2. Set up your environment:
   a. Set up environment variables for use with a particular installation by entering one the following command:
      ```
      . MQ_INSTALLATION_PATH/bin/setmqenv -s
      ```
      where *MQ_INSTALLATION_PATH* refers to the location where IBM MQ is installed.
   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 a queue manager called `QMA` by entering the following command:
   ```
   crtmqm QMA
   ```
   Messages indicate when the queue manager is created, and when the default IBM MQ objects are created.
4. Start the queue manager by entering the following command:
   ```
   strmqm QMA
   ```
   A message indicates when the queue manager starts.
5. Start MQSC by entering the following command:
   ```
   runmqsc QMA
   ```
   A message indicates when MQSC starts. MQSC has no command prompt.
6. Define a local queue called `QUEUE1` by entering the following command:
   ```
   DEFINE QLOCAL (QUEUE1)
   ```

A message indicates when the queue is created.

7. Stop MQSC by entering the following command:

```
end
```

Messages are shown, followed by the command prompt.

Note: Subsequent steps require that the samples package is installed.

8. Change into the *MQ_INSTALLATION_PATH*/samp/bin directory, which contains the sample programs. *MQ_INSTALLATION_PATH* represents the high-level directory in which IBM MQ is installed.
9. Put a message on the queue by entering the following commands

```
./amqsput QUEUE1 QMA
```

The following messages are shown:

```
Sample AMQSPUT0 start
target queue is QUEUE1
```

10. Type some message text on one or more lines, where each line is a different message. 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. Get the messages from the queue, by entering the following command:

```
./amqsget QUEUE1 QMA
```

The sample program starts, and your messages are displayed.

Results

You have successfully verified your local installation.

Verifying a local server installation using the Postcard application on Solaris: ▶ Solaris

Sending messages successfully between two Postcard applications verifies a local installation.

Before you begin

The postcard application is Java based and requires a system with the ability to view a graphical display.

You must ensure that you are a member of the IBM MQ administrators group (**mqm**).

Note: Using Postcard to verify an IBM MQ installation is only possible if there is one IBM MQ installation on that box. The Default Configuration wizard will not create a default configuration if a queue manager already exists on the box. The Default Configuration wizard will run on any installation on a box but only one default configuration can be created per box. Using Postcard to verify second and subsequent installations of IBM MQ on the same box is not possible.

To verify that the local installation is working, you can run two instances of the Postcard application on the same server. The postcard application can send messages to, and receive messages from, other postcard applications. Successful sending and receiving of messages verifies that IBM MQ is installed and working correctly on the server.

Procedure

1. Log on as a user in group **mqm**.
2. Start the postcard application in one of the following ways:

a. From the command line:
   1) Change the directory to `MQ_INSTALLATION_PATH/java/bin`. `MQ_INSTALLATION_PATH` represents the high-level directory in which IBM MQ is installed.
   2) Run the postcard application by entering the following command:
      ```
      ./postcard
      ```
b. From the IBM MQ Explorer:
   1) If the Welcome to IBM MQ Explorer Content view page does not show, click **IBM MQ** in the Navigator view to show the Welcome page.
   2) Click **Launch Postcard** to start the Postcard.

3. At the Postcard - Sign On window, type in a nickname to use to send messages within the Postcard application (for example, `User1`).
4. Select the queue manager to use as the mailbox:
   - If you do not have any queue managers, you are prompted to either launch the Default Configuration or close the Postcard application. Launching the Default Configuration creates a default queue manager.
   - If the only queue manager on your server is the default queue manager, this queue manager is used automatically for the postcard application. The default queue manager is created by running the Default Configuration wizard
   - If you have created your own queue managers, but you have not run the Default Configuration wizard, select an appropriate queue manager from the list.
   - If you have run the Default Configuration wizard and you want to use the default queue manager, but there are other queue managers on your server, select the **Advanced** check box. Then select **Use Default Configuration as mailbox**.
   - If you have run the Default Configuration wizard and also created your own queue managers, and you do not want to use the default queue manager, select the **Advanced** check box. Then select **Choose queue manager as mailbox**, and then select the appropriate queue manager from the list.

   When your selection is complete, click **OK** to display your first Postcard window.
5. Run a second instance of the Postcard application by following the steps used to open the first instance of the Postcard application.
6. The Postcard - Sign On panel is displayed again. Type in a second nickname to use to send messages within this second Postcard application (for example, `User2`).
7. Repeat the selection of the queue manager that you want to use as the mailbox (as described in step 4). The queue manager you select for this second Postcard must be the same queue manager as used for the first instance of the Postcard application.
8. In the first Postcard, (User1), enter the nickname (`User2`) for the second Postcard application in the **To:** field. Because the sender and receiver are on the same server, you can leave the **On:** field blank.
9. Type a message in the **Message:** field and click **Send**.
10. The **Postcards sent and received** area of the Postcard shows details of the message. In the sending Postcard, the message is displayed as sent. In the receiving Postcard, the message is displayed as received.
11. In the receiving Postcard, (User2), double-click the message in the **Postcards sent and received** area to view it. When this message arrives, it verifies that IBM MQ is correctly installed.

What to do next

Depending on your situation, you might want to do the following tasks:
- Install IBM MQ on other servers. Follow the installation procedure for the appropriate platform. Ensure that you use the **Join Default Cluster** window in the Default Configuration wizard to add the other servers to the cluster on your first server.

- Install the IBM MQ MQI client on other servers.
- Continue with further administration tasks, see Administering IBM MQ.

Verifying a server-to-server installation on Solaris

Solaris

You can use either the command line or the postcard application to verify a server-to-server installation on Solaris.

Before you begin

For a server-to-server verification, the communication links between the two systems must be checked. Before you can do the verification, you must therefore ensure that the communications protocol is installed and configured on both systems.

On Solaris, IBM MQ supports both TCP and SNA.

The examples in this task use TCP/IP. If you do not use TCP, see Setting up communication on UNIX and Linux.

About this task

For a server-to server installation, you can use the command line to verify that IBM MQ is successfully installed, and that the associated communication links are working properly.

You can also verify an installation using the postcard application. The postcard application is Java based and requires a system with the ability to view a graphical display.

Procedure

- To use the command line to verify an installation, see "Verifying a server-to-server installation using the command line on Solaris."
- To use the postcard application to verify an installation, see "Verifying a server-to-server installation using the Postcard application on Solaris" on page 427.

Verifying a server-to-server installation using the command line on Solaris: Solaris

You can verify a server-to-server installation using two servers, one as a sender and one as a receiver.

Before you begin

- Make sure that TCP/IP and IBM MQ are installed on both servers (see "Verifying a server-to-server installation on Solaris").
- Make sure that you are a member of the IBM MQ administrators group (**mqm**) on each server.
- Decide which installation is the sender server and which installation is the receiver server. The installations might be on the same system, or on different systems.

About this task

IBM MQ object definitions are case-sensitive. Any text entered as an MQSC command in lowercase is converted automatically to uppercase unless you enclose it in single quotation marks. Make sure that you type the examples exactly as shown.

Procedure

1. On the **receiver** server:
   a. On AIX, log in as a user in the mqm group.
   b. Check which ports are free, for example by running **netstat**. For more information about this command, see the documentation of your operating system.

      If port 1414 is not in use, make a note of 1414 to use as the port number in step 2 h. Use the same number for the port for your listener later in the verification. If it is in use, note a port that is not in use; for example 1415.
   c. Set up the environment for the installation you are using by entering the following command at the command prompt:

      ```
      . MQ_INSTALLATION_PATH/bin/setmqenv -s
      ```

      where *MQ_INSTALLATION_PATH* refers to the location where IBM MQ is installed.
   d. Create a queue manager called QMB by entering the following command at the command prompt:

      ```
      crtmqm QMB
      ```

      Messages tell you that the queue manager has been created, and that the default IBM MQ objects have been created.
   e. Start the queue manager by entering the following command:

      ```
      strmqm QMB
      ```

      A message tells you when the queue manager has started.
   f. Start MQSC by entering the following command:

      ```
      runmqsc QMB
      ```

      A message tells you that MQSC has started. MQSC has no command prompt.
   g. Define a local queue called RECEIVER.Q by entering the following command:

      ```
      DEFINE QLOCAL (RECEIVER.Q)
      ```

      A message tells you the queue has been created.
   h. Define a listener by entering the following command:

```
DEFINE LISTENER (LISTENER1) TRPTYPE (TCP) CONTROL (QMGR) PORT ( PORT_NUMBER )
```

      Where *port_number* is the name of the port the listener runs on. This number must be the same as the number used when defining your sender channel.
   i. Start the listener by entering the following command:

      ```
      START LISTENER (LISTENER1)
      ```

      Note: Do not start the listener in the background from any shell that automatically lowers the priority of background processes.
   j. Define a receiver channel by entering the following command:

      ```
      DEFINE CHANNEL (QMA.QMB) CHLTYPE (RCVR) TRPTYPE (TCP)
      ```

      A message tells you when the channel has been created.
   k. End MQSC by typing:

      ```
      end
      ```

      Some messages are displayed, followed by the command prompt.
2. On the **sender** server:
   a. As the sender server is an AIX system, log in as a user in the mqm group.

b. Set up the environment for the installation you are using by entering the following command at the command prompt:

```
. MQ_INSTALLATION_PATH/bin/setmqenv -s
```

where `MQ_INSTALLATION_PATH` refers to the location where IBM MQ is installed.

c. Create a queue manager called QMA by entering the following command at the command prompt:

```
crtmqm QMA
```

Messages tell you that the queue manager has been created, and that the default IBM MQ objects have been created.

d. Start the queue manager, by entering the following command:

```
strmqm QMA
```

A message tells you when the queue manager has started.

e. Start MQSC by entering the following command:

```
runmqsc QMA
```

A message tells you that an MQSC session has started. MQSC had no command prompt.

f. Define a local queue called QMB (to be used as a transmission queue) by entering the following command:

```
DEFINE QLOCAL (QMB) USAGE (XMITQ)
```

A message tells you when the queue has been created.

g. Define a local definition of the remote queue with by entering the following command:

```
DEFINE QREMOTE (LOCAL.DEF.OF.REMOTE.QUEUE) RNAME (RECEIVER.Q) RQMNAME ('QMB') XMITQ (QMB)
```

h. Define a sender channel by entering one of the following commands:

con-name is the TCP/IP address of the receiver system. If both installations are on the same system, the *con-name* is `localhost`. *port* is the port you noted in 1 b. If you do not specify a port, the default value of 1414 is used.

```
DEFINE CHANNEL (QMA.QMB) CHLTYPE (SDR) CONNAME ('CON-NAME(PORT)') XMITQ (QMB) TRPTYPE (TCP)
```

i. Start the sender channel by entering the following command:

```
START CHANNEL(QMA.QMB)
```

The receiver channel on the receiver server starts automatically when the sender channel starts.

j. Stop MQSC by entering the following command:

```
end
```

Some messages are displayed, followed by the command prompt.

k. If the sender server is a UNIX or Linux system, change into the `MQ_INSTALLATION_PATH/samp/bin` directory. This directory contains the sample programs. `MQ_INSTALLATION_PATH` represents the high-level directory in which IBM MQ is installed.

l. If both the sender server and receiver server are installations on the same system, check that the queue managers have been created on different installations by entering the following command:

```
dspmq -o installation
```

If the queue managers are on the same installation, move either QMA to the sender installation or QMB to the receiver installation by using the **setmqm** command. For more information, see setmqm.

m. Put a message on the local definition of the remote queue, which in turn specifies the name of the remote queue. Enter one of the following commands:

- On Windows:

```
amqsput LOCAL.DEF.OF.REMOTE.QUEUE QMA
```

- On UNIX and Linux:

  ```
  ./amqsput LOCAL.DEF.OF.REMOTE.QUEUE QMA
  ```

  A message tells you that `amqsput` has started.

n. Type some message text on one or more lines, followed by a blank line. A message tells you that `amqsput` has ended. Your message is now on the queue and the command prompt is displayed again.

3. On the **receiver** server:

   a. As your receiver server is an AIX system, change into the *MQ_INSTALLATION_PATH*/samp/bin directory. This directory contains the sample programs. *MQ_INSTALLATION_PATH* represents the high-level directory in which IBM MQ is installed.

   b. Get the message from the queue on the receiver by entering the following command:

   ```
   ./amqsget RECEIVER.Q QMB
   ```

   The sample program starts, and your message is displayed. After a pause, the sample ends. Then the command prompt is displayed.

Results

You have now successfully verified the server-to-server installation.

Verifying a server-to-server installation using the Postcard application on Solaris: Solaris

You can use two instances of the Postcard application to verify that a server-to-server installation is working.

Before you begin

You can use the Postcard application on two servers, one instance of the Postcard application on each server, to verify that a server-to-server installation is working. Successful sending and receiving of messages verifies that IBM MQ is successfully installed, and that communication between the two servers is working correctly.

Note:
- If the system has multiple IBM MQ installations, ensure that Postcard has not been run before on any installations on that server. As the default configuration can only exist on one IBM MQ installation per system, the Default Configuration wizard and Postcard can not be used for verification of a second or any subsequent installation.
- The two server installations must be on different systems to do a server-to-server verification using the postcard application. To verify a server-to-server installation on the same machine, you can use the command line.
- Make sure that TCP/IP and IBM MQ are installed on both servers.
- Make sure that your systems are able to view a graphical display.
- Make sure that you are a member of the IBM MQ administrators group (**mqm**) on each server.
- Check that one of the following scenarios applies:
  - Neither server has had any queue managers created.
  - Use the Default Configuration wizard to create default queue managers on each server and link them to the default cluster.

    Details on how to use the Default Configuration wizard are provided in this topic.
  - Both servers have existing queue managers and these queue managers are in the same cluster.

If your queue managers are not in the same cluster, create new queue managers on both servers. Then create a cluster, and ensure that the queue managers that you create on each server belong to that cluster.

– You have configured channels to communicate between the two servers.

  For instructions on how to set up the channels, see “Verifying a server-to-server installation using the command line on Solaris” on page 424. After you have set up the channels, follow the instructions in this topic to verify your server-to-server installation.

Procedure

1. On the first server, log on as a user in group **`mqm`**.
2. Start the postcard application in one of the following ways:
   a. From the command line:
      1) Change the directory to *`MQ_INSTALLATION_PATH`*`/java/bin`. *`MQ_INSTALLATION_PATH`* represents the high-level directory in which IBM MQ is installed.
      2) Run the postcard application by entering the following command:
         ```
         ./postcard
         ```
   b. From the IBM MQ Explorer:
      1) If the Welcome to IBM MQ Explorer Content view page does not show, click **IBM MQ** in the Navigator view to show the Welcome page.
      2) Click **Launch Postcard** to start the Postcard.
3. At the Postcard - Sign On window, type a nickname to use to send messages within the Postcard application. For example, `User1` for the first server, and `User2` for the second server.
4. When you have completed the wizard, you are taken back to the Postcard - Sign On window.
5. Select the queue manager to use as the mailbox:
   - If you do not have any queue managers, you are prompted to either launch the Default Configuration or close the Postcard application. Work through the Default Configuration wizard. When you get to the option to join the queue manager to the default cluster, tick the check box. On the next screen:
     – For the first server, select **yes, make it the repository for the cluster**.
     – For the second server, select **No another computer has already joined the cluster as a repository**. When requested, enter the location of the repository, by typing the name of the sender server.
   - If the only queue manager on your server is the default queue manager, this queue manager is used automatically for the postcard application. The default queue manager is created by running the Default Configuration wizard
   - If you have created your own queue managers, but you have not run the Default Configuration wizard, select an appropriate queue manager from the list.
   - If you have run the Default Configuration wizard and you want to use the default queue manager, but there are other queue managers on your server, select the **Advanced** check box. Then select **Use Default Configuration as mailbox**.
   - If you have run the Default Configuration wizard and also created your own queue managers, and you do not want to use the default queue manager, select the **Advanced** check box. Then select **Choose queue manager as mailbox**, and then select the appropriate queue manager from the list.

   When your selection is complete, click **OK**.
6. Complete steps 1 - 5 for the second server.
7. In the Postcard on the first server:
   a. Enter the nickname (`user2`) for the Postcard application on the second server in the **To:** field.
   b. Enter the queue manager on the second server in the **On:** field.
   c. Type a message in the **Message:** field and click **Send**.

8. In the Postcard on the second server:
   a. In the **Postcards sent and received**, double-click the message marked as received to view the message from the first server.
   b. Optional: Send a postcard to the first server by adapting the instructions in step 7. You must enter details of the first server in the **To:** field and the **On:** field.

   The messages verify that IBM MQ is correctly installed and that your communication link between the two servers is working correctly.

Verifying a client installation using the command line on Solaris

Solaris

You can verify a client installation using the command line. On the server you create a queue manager, a local queue, a listener, and a server-connection channel. You must also apply security rules to allow the client to connect and make use of the queue defined. On the client you create a client-connection channel, and then use the sample PUT and GET programs to complete the verification procedure.

The verification procedure shows how to create a queue manager called `queue.manager.1`, a local queue called `QUEUE1`, and a server-connection channel called `CHANNEL1` on the server.

It shows how to create the client-connection channel on the IBM MQ MQI client workstation. It then shows how to use the sample programs to put a message onto a queue, and get the message from the queue.

The example does not address any client security issues. See Setting up IBM MQ MQI client security for details if you are concerned with IBM MQ MQI client security issues.

The verification procedure assumes that:
- The full IBM MQ server product has been installed on a server.
- The server installation is accessible on your network.
- The IBM MQ MQI client software has been installed on a client system.
- The IBM MQ sample programs have been installed.
- TCP/IP has been configured on the server and client systems. For more information, see Configuring connections between the server and client.

First, set up the server using the command line, using the instructions in "Setting up the server using the command line on Solaris" on page 430.

Once you have set up the server, you must set up the client, using the instructions in "Connecting to a queue manager, using the `MQSERVER` environment variable on Solaris" on page 431.

Finally, you can test the communications between client and server, using the instructions in "Testing communication between a client and a server on Solaris" on page 432.

Setting up the server using the command line on Solaris: ▶ Solaris

Follow these instructions to create a queue manager, queue, and channel on the server. You can then use these objects to verify the installation.

About this task

These instructions assume that no queue manager or other IBM MQ objects have been defined.

IBM MQ object definitions are case-sensitive. Any text entered as an MQSC command in lowercase is converted automatically to uppercase unless you enclose it in single quotation marks. Make sure that you type the examples exactly as shown.

Procedure

1. Create a user ID on the server that is not in the `mqm` group. This user ID must exist on the server and client. This is the user ID that the sample applications must be run as, otherwise a 2035 error is returned.
2. Log in as a user in the mqm group.
3. You must set various environment variables so that the installation can be used in the current shell. You can set the environment variables by entering the following command:
```
. MQ_INSTALLATION_PATH/bin/setmqenv -s
```
   where *`MQ_INSTALLATION_PATH`* refers to the location where IBM MQ is installed.
4. Create a queue manager called `QUEUE.MANAGER.1` by entering the following command:
```
crtmqm QUEUE.MANAGER.1
```
   You see messages telling you that the queue manager has been created.
5. Start the queue manager by entering the following command:
```
strmqm QUEUE.MANAGER.1
```
   A message tells you when the queue manager has started.
6. Start MQSC by entering the following command:
```
runmqsc QUEUE.MANAGER.1
```
   A message tells you that an MQSC session has started. MQSC has no command prompt.
7. Define a local queue called `QUEUE1` by entering the following command:
```
DEFINE QLOCAL(QUEUE1)
```
   A message tells you when the queue has been created.
8. Allow the user ID that you created in step 1 to use `QUEUE1` by entering the following command:
```
SET AUTHREC PROFILE(QUEUE1) OBJTYPE(QUEUE) PRINCIPAL(' non_mqm_user ') AUTHADD(PUT,GET)
```
   where *non_mqm_user* is the user ID created in step 1. A message tells you when the authorization has been set. You must also run the following command to give the user ID authority to connect:
```
SET AUTHREC OBJTYPE(QMGR) PRINCIPAL(' non_mqm_user ') AUTHADD(CONNECT)
```
   If this command is not run, a 2305 stop error is returned.
9. Define a server-connection channel by entering the following command:
```
DEFINE CHANNEL (CHANNEL1) CHLTYPE (SVRCONN) TRPTYPE (TCP)
```
   A message tells you when the channel has been created.

10. Allow your client channel to connect to the queue manager and run under the user ID that you created in step 1, by entering the following MQSC command:
    ```
    SET CHLAUTH(CHANNEL1) TYPE(ADDRESSMAP) ADDRESS(' client_ipaddr ') MCAUSER(' non_mqm_user ')
    ```
    where *client_ipaddr* is the IP address of the client system, and *non_mqm_user* is the user ID created in step 1. A message tells you when the rule has been set.
11. Define a listener by entering the following command:
    ```
    DEFINE LISTENER (LISTENER1) TRPTYPE (TCP) CONTROL (QMGR) PORT (port_number)
    ```
    where *port_number* is the number of the port the listener is to run on. This number must be the same as the number used when defining your client-connection channel in "Installing an IBM MQ client on Solaris" on page 416.

    Note: If you omit the port parameter from the command, a default value of 1414 is used for the listener port. If you want to specify a port other than 1414, you must include the port parameter in the command, as shown.
12. Start the listener by entering the following command:
    ```
    START LISTENER (LISTENER1)
    ```
13. Stop MQSC by entering:
    ```
    end
    ```
    You see some messages, followed by the command prompt.

What to do next

Follow the instructions to set up the client. See "Connecting to a queue manager, using the MQSERVER environment variable on Solaris."

Connecting to a queue manager, using the MQSERVER environment variable on Solaris: Solaris

When an IBM MQ application is run on the IBM MQ MQI client, it requires the name of the MQI channel, the communication type, and the address of the server to be used. Provide these parameters by defining the MQSERVER environment variable.

Before you begin

Before you start this task, you must complete the task, "Setting up the server using the command line on Solaris" on page 430, and save the following information:
- The host name or IP address of the server and port number that you specified when creating the listener.
- The channel name of the server-connection channel.

About this task

This task describes how to connect an IBM MQ MQI client, by defining the MQSERVER environment variable on the client.

You can give the client access to the generated client channel definition table, amqclchl.tab instead; see Accessing client-connection channel definitions.

Procedure
1. Log in as the userid that you created in Step 1 of "Setting up the server using the command line on Solaris" on page 430.

2. Check the TCP/IP connection. From the client, enter one of the following commands:
   - `ping server-hostname`
   - `ping n.n.n.n`

     `n.n.n.n` represents the network address. You can set the network address in IPv4 dotted decimal form, for example, `192.0.2.0`. Alternatively, set the address in IPv6 hexadecimal form, for example `2001:0DB8:0204:acff:fe97:2c34:fde0:3485`.

   If the **`ping`** command fails, correct your TCP/IP configuration.
3. Set the `MQSERVER` environment variable. From the client, enter the following command:
   ```
   export MQSERVER=CHANNEL1/TCP/' server-address (port)'
   ```
   Where:
   - *CHANNEL1* is the server-connection channel name.
   - *server-address* is the TCP/IP host name of the server.
   - *port* is the TCP/IP port number the server is listening on.

   If you do not give a port number, IBM MQ uses the one specified in the `qm.ini` file, or the client configuration file. If no value is specified in these files, IBM MQ uses the port number identified in the TCP/IP services file for the service name `MQSeries`. If an `MQSeries` entry in the services file does not exist, a default value of `1414` is used. It is important that the port number used by the client and the port number used by the server listener program are the same.

What to do next

Use the sample programs to test communication between the client and server; see "Testing communication between a client and a server on Solaris."

Testing communication between a client and a server on Solaris: Solaris

On the IBM MQ MQI client workstation, use the amqsputc sample program to put a message on the queue at the server workstation. Use the amqsgetc sample program to get the message from the queue back to the client.

Before you begin

Complete the previous topics in this section:
- Set up a queue manager, channels, and queue.
- Open a command window.
- Set system environment variables.

About this task

Note that IBM MQ object definitions are case-sensitive. Text entered as an MQSC command in lowercase is converted automatically to uppercase unless you enclose it in single quotation marks. Make sure that you type the examples exactly as shown.

Procedure
1. Change to the `MQ_INSTALLATION_PATH/samp/bin directory`, which contains the sample programs. *`MQ_INSTALLATION_PATH`* represents the high-level directory in which IBM MQ is installed.
2. You must set certain environment variables so that the installation can be used in the current shell. You can set the environment variables by entering the following command:
   ```
   . MQ_INSTALLATION_PATH/bin/setmqenv -s
   ```
   where *`MQ_INSTALLATION_PATH`* refers to the location where IBM MQ is installed.

3. Start the PUT program for `QUEUE1` on `QUEUE.MANAGER.1` by entering the following command:

   ```
   ./amqsputc QUEUE1 QUEUE.MANAGER.1
   ```

   If the command is successful, the following messages are displayed:

   ```
   Sample AMQSPUT0 start target queue is QUEUE1
   ```

   Tip: You might get the error, `MQRC_NOT_AUTHORIZED` (2035). By default, channel authentication is enabled when a queue manager is created. Channel authentication prevents privileged users accessing a queue manager as an IBM MQ MQI client. For verifying the installation, you can either change the MCA user ID to a non-privileged user, or disable channel authentication. To disable channel authentication run the following MQSC command:

   ```
   ALTER QMGR CHLAUTH(DISABLED)
   ```

   When you finish the test, if you do not delete the queue manager, re-enable channel authentication:

   ```
   ALTER QMGR CHLAUTH(ENABLED)
   ```
4. Type some message text, then press **Enter** twice. The following message is displayed:

   ```
   Sample AMQSPUT0 end
   ```

   Your message is now on the queue that is on the server queue manager.
5. Start the GET program for `QUEUE1` on `QUEUE.MANAGER.1` by entering the following command:

   ```
   ./amqsgetc QUEUE1 QUEUE.MANAGER.1
   ```

   The sample program starts, and your message is displayed. After a short pause (approximately 30 seconds), the sample ends and the command prompt is displayed again.

Results

You have now successfully verified the client installation.

What to do next

1. You must set various environment variables on the server so that the installation can be used in the current shell. You can set the environment variables by entering the following command:

   ```
   . MQ_INSTALLATION_PATH/bin/setmqenv -s
   ```

   where *`MQ_INSTALLATION_PATH`* refers to the location where IBM MQ is installed.
2. On the server, stop the queue manager by entering the following command:

   ```
   endmqm QUEUE.MANAGER.1
   ```
3. On the server, delete the queue manager by entering the following command:

   ```
   dltmqm QUEUE.MANAGER.1
   ```

Uninstalling IBM MQ on Solaris

▶ Solaris

On Solaris, you can uninstall the IBM MQ server or client using the **pkgrm** command.

Before you begin

If any updates have been applied, remove them before starting this uninstallation procedure. For more information, see Restoring the previous maintenance level on IBM MQ on Solaris.

Restriction: On Solaris, you cannot remove components from an installation. There is no supported method of doing this.

Important: You must stop all IBM MQ queue managers, other objects, and applications, before you begin the process to uninstall or modify IBM MQ.

Procedure

1. Stop all IBM MQ applications associated with the installation you are uninstalling or modifying, if you have not already done so.
2. For a server installation, end any IBM MQ activity associated with the installation you are uninstalling:
   a. Log in as a user in the group `mqm`.
   b. Set up your environment to work with the installation you want to uninstall. Enter the following command:
      ```
      . MQ_INSTALLATION_PATH/bin/setmqenv
      ```
      where *MQ_INSTALLATION_PATH* refers to the location where IBM MQ is installed.
   c. Display the state of all queue managers on the system. Enter the following command:
      ```
      dspmq
      ```
   d. Stop all running queue managers associated with the installation you want to uninstall. Enter the following command for each queue manager:
      ```
      endmqm QMgrName
      ```
   e. Stop any listeners associated with the queue managers. Enter the following command for each queue manager:
      ```
      endmqlsr -m QMgrName
      ```
3. Log in as root.
4. Uninstall IBM MQ using **pkgrm**:
   a. On a system with a single installation, enter the following command:
      ```
      pkgrm mqm
      ```
   b. On a system with multiple installations:
      ```
      pkgrm mqm-suffix
      ```
      where *suffix* is the unique name given to the packages when **crtmqpkg** was run at installation time. *suffix* is included in each of the package names that belong to a particular installation. The first installation on the system does not have a *suffix*, and is uninstalled using the same method as for a single installation.

   If a package has a dependency on `mqm`, **pkgrm** returns the name of the package. Uninstall the dependent packages first.

Results

After uninstallation, certain files under the directory trees `/var/mqm` and `/etc/opt/mqm` are not removed. These files contain user data and remain so subsequent installations can reuse the data. Most of the remaining files contain text, such as INI files, error logs, and FDC files. The directory tree `/var/mqm/shared` contains files that are shared across installations, including the executable shared library `libmqzsd.so`.

What to do next

- If the product successfully uninstalled, you can delete any files and directories contained in the installation directory.
- If there are no other IBM MQ installations on the system, and you are not planning to reinstall or migrate, you can delete the `/var/mqm` and `/etc/opt/mqm` directory trees, including the file `libmqzsd.so`. Deleting these directories destroys all queue managers and their associated data.

Installing and uninstalling IBM MQ on Windows

Windows

Installation tasks that are associated with installing IBM MQ on Windows systems are grouped in this section.

About this task

To prepare for installation and to install the IBM MQ components, complete the following tasks.

For information about how to uninstall IBM MQ, see “Uninstalling IBM MQ on Windows” on page 514.

If product fixes or updates are made available, see IBM MQ maintenance tasks for information about how to apply these changes.

Procedure

1. Check the system requirements. See “Checking requirements on Windows” on page 444.
2. Plan your installation.
   - As part of the planning process, you must choose which components to install and where to install them. See “IBM MQ features for Windows systems” on page 436.
   - You must also make some platform-specific choices. See “Planning to install IBM MQ on Windows” on page 446.
3. Install IBM MQ server. See “Installing IBM MQ server on Windows” on page 453.
4. Optional: Install an IBM MQ client. See “Installing an IBM MQ client on Windows” on page 481.
5. Verify your installation. See “Verifying an IBM MQ installation on Windows” on page 496.

Related information:

Video: Installing and verifying IBM MQ V9.0 (YouTube)

IBM MQ features for Windows systems

Windows

You can select the features that you require when you install IBM MQ.

Important: See IBM MQ license information for details of what each purchase of IBM MQ entitles you to install.

If you choose an interactive installation, before you install, you must decide what type of installation you require. For more information about the available types of installation and the features that are installed with each option, see "Installation methods for Windows" on page 448.

The following table shows the features that are available when installing an IBM MQ server or client on a Windows system.

Interactive displayed name	Non-interactive displayed name	Description	Server media	Client media
Server	Server	You can use the server to run queue managers on your system and connect to other systems over a network. Provides messaging and queuing services to applications, and support for IBM MQ client connections. V 9.0.2 From IBM MQ Version 9.0.2, additional prerequisite checking is performed on this	✓	
IBM MQ Explorer	Explorer	IBM MQ Explorer allows you to administer and monitor resources in IBM MQ.	✓	

Interactive displayed name	Non-interactive displayed name	Description	Server media	Client media
Managed File Transfer Service	MFT Service	The Managed File Transfer Service install option installs a file transfer agent that has additional capabilities beyond those provided by the file transfer agent installed using the Managed File Transfer Agent install option. These additional capabilities are:- • Create protocol bridge agents which are used to send and receive files with legacy FTP, FTPS or SFTP servers The Managed File Transfer Service install option must be installed on systems where the IBM MQ Server install option is already installed.	✓	
Managed File Transfer Logger	MFT Logger	The Managed File Transfer Logger install option installs a file transfer logger which connects to an IBM MQ queue manager, often the queue manager designated as the coordination queue manager. It logs file transfer audit related data to either a database or a file. It must be installed on systems where the IBM MQ Server install option is already installed.	✓	

Interactive displayed name	Non-interactive displayed name	Description	Server media	Client media
Managed File Transfer Agent	MFT Agent	The Managed File Transfer Agent install option installs a file transfer agent which connects to an IBM MQ queue manager and transfers file data, as messages, to other file transfer agents. These must be installed either as part of the Managed File Transfer Agent or Managed File Transfer Service install options.	✓	
Managed File Transfer Tools	MFT Tools	The Managed File Transfer Tools install option installs command line tools that are used to interact with file transfer agents. You can use these tools to start file transfers, schedule file transfers and create resource monitors from the command line. The Managed File Transfer Tools can be installed and used on either a system where file transfer agents are installed, or on a system where no file transfer agents are installed.	✓	

Interactive displayed name	Non-interactive displayed name	Description	Server media	Client media
LTS Windows Client V 9.0.1 MQI Client	Client	The Windows client is a small subset of IBM MQ, without a queue manager, that uses the queue manager and queues on other (server) systems. It can be used only when the system it is on is connected to another system that is running a full server version of IBM MQ. The client and server can be on the same system if required. V 9.0.1 From Version 9.0.1, this feature is named MQI Client.	✓	✓
LTS Java and .NET Messaging and Web Services V 9.0.1 Extended Messaging APIs	JavaMsg	The files needed for messaging using Java. This feature includes support for JMS, XMS, .NET, and IBM MQ web services. V 9.0.1 From Version 9.0.1, this feature is named Extended Messaging APIs.	✓	✓
V 9.0.1 Web Administration	V 9.0.1 Web	V 9.0.1 Adds HTTP based administration for IBM MQ through the REST API and IBM M If you want to install the Web Administration feature you must also install the I	✓	

Interactive displayed name	**Non-interactive displayed name**	**Description**	**Server media**	**Client media**
Development Toolkit	Toolkit	This feature includes sample source files, and the bindings (files .H, .LIB, .DLL, and others), that you need to develop applications to run on IBM MQ. Bindings and samples are provided for the following languages: C, C++, Visual Basic, ActiveX, Cobol, and .NET (including C#). Java and Java Message Service support is included and samples are provided for MTS (COM+), and MQSC.	✓	✓

Interactive displayed name	Non-interactive displayed name	Description	Server media	Client media
Telemetry Service	XR Service	MQ Telemetry supports the connection of Internet Of Things (IOT) devices (that is, remote sensors, actuators and telemetry devices) that use the IBM MQ Telemetry Transport (MQTT) protocol. The telemetry service, which is also know as the MQXR service, enables a queue manager to act as an MQTT server, and communicate with MQTT client apps. A set of MQTT client libraries is also available in the IBM Messaging Telemetry Clients SupportPac. These libraries help you write the MQTT client apps that IOT devices use to communicate with MQTT servers.The XR Service install option must be installed on systems where the IBM MQ Server install option is already installed. See also "Installation considerations for MQ Telemetry" on page 538.	✓	

Interactive displayed name	Non-interactive displayed name	Description	Server media	Client media
Advanced Message Security	AMS	Provides a high level of protection for sensitive data flowing through the IBM MQ network, while not impacting the end applications. You must install this component on all IBM MQ installations that host queues you want to protect. You must install the IBM Global Security Kit component on any IBM MQ installation that is used by a program that puts or gets messages to or from a protected queue, unless you are using only Java client connections.The AMS install option must be installed on systems where the IBM MQ Server install option is already installed.	✓	
AMQP Service	AMQP	Install this component to make AMQP channels available. AMQP channels support MQ Light APIs. You can use AMQP channels to give AMQP applications access to the enterprise-level messaging facilities provided by IBM MQ. The AMQP Service install option must be installed on systems where the IBM MQ Server install option is already installed.	✓	

Interactive displayed name	Non-interactive displayed name	Description	Server media	Client media
V 9.0.2 Java Runtime Environment	JRE	The Java Runtime Environment (JRE) has become a separate feature at IBM MQ Version 9.0.2. The JRE feature installs a JRE that has been tailored for IBM MQ use, and is a required feature for all other features that use Java. That is: • IBM MQExplorer • Web Administration • Telemetry Service • AMQP Service • Managed File Transfer Additional prerequisite checking is performed on this option. See Prerequisite checking for more information.	✓	✓

V 9.0.2

Windows standard installation features

The following features are part of the Windows standard installation feature set. They are the features installed by the GUI installer for a "*typical installation*".

Interactive displayed name	Non-interactive displayed name	Notes
Server	Server	
MQ Explorer	Explorer	
Java and .NET messaging and Web Services (Version 9.0.0) Extended Messaging APIs (Version 9.0.1)	JavaMsg	Feature renamed at Version 9.0.1
Web Administration (Version 9.0.1)	Web	Feature added at Version 9.0.1
Development Toolkit	Toolkit	
V 9.0.2 Java Runtime Environment	JRE	Feature added at Version 9.0.2. Prior to Version 9.0.2 JRE was always installed.

When you install an IBM MQ server, using **msiexec**, the features that are included in a *typical installation* are added to the list of features you specify in the **ADDLOCAL** directive.

If you specify **ADDLOCAL**="" all these features will be installed.

If you do not want specific features added, you must add those specific features to the **REMOVE** directive.

For example, suppose that you specify the following settings for an **msiexec** installation:

```
ADDLOCAL="Client"
REMOVE="Web,Toolkit"
```

This results in the following features being installed:

```
Server,Explorer,JavaMsg,JRE,Client
```

Related concepts:

"IBM MQ components and features" on page 192
You can select the components or features that you require when you install IBM MQ.

"Planning considerations for installation on Multiplatforms" on page 196
Before you install IBM MQ, you must choose which components to install and where to install them. You must also make some platform-specific choices.

Checking requirements on Windows

▶ Windows

Before you install IBM MQ on Windows, you must check for the latest information and system requirements.

About this task

A summary of the tasks that you must complete to check system requirements are listed here with links to further information.

Procedure

1. Check that you have the latest information, including information on hardware and software requirements. See "Where to find product requirements and support information" on page 195.
2. Check that your systems meet the initial hardware and software requirements for Windows. See "Hardware and software requirements on Windows systems" on page 445.

   The supported hardware and software environments are occasionally updated. See System Requirements for IBM MQ for the latest information.
3. Check that your systems have sufficient disk space for the installation. See Disk space requirements.
4. Check that you have the correct licenses. See "License requirements" on page 194 and IBM MQ license information.

Related concepts:

"IBM MQ installation overview" on page 192
An overview of concepts and considerations for installing IBM MQ, with links to instructions on how to install, verify, and uninstall IBM MQ on each of the supported platforms.

Related information:

IBM MQ maintenance tasks

Hardware and software requirements on Windows systems

Windows

Check that the server environment meets the prerequisites for installing IBM MQ for Windows and install any prerequisite software that is missing from your system from the server DVD.

Before you install IBM MQ, you must check that your system meets the hardware and software requirements. For the latest details of hardware and software requirements on all supported platforms, see System Requirements for IBM MQ.

You must also review the product readme file, which includes information about last-minute changes and known problems and workarounds. For the latest version of the product readme file, see the IBM MQ, WebSphere MQ, and MQSeries product readmes web page.

Storage requirements for IBM MQ server

The storage requirements depend on which components you install, and how much working space you need. The storage requirements also depend on the number of queues that you use, the number and size of the messages on the queues, and whether the messages are persistent. You also require archiving capacity on disk, tape, or other media. For more information, see System Requirements for IBM MQ.

Disk storage is also required:

- Prerequisite software
- Optional software
- Your application programs

Requirements for IBM MQ Explorer

IBM MQ Explorer can be installed either as part of the product installation, or from the stand-alone IBM MQ Explorer support pack MS0T.

- The product version is available for Windows x86_64.
- The support pack version is available for Windows x86 and x86_64.

The requirements for installing IBM MQ Explorer as part of the product installation, and not as the stand-alone IBM MQ Explorer support pack MS0T, include:

- A 64-bit (x86_64) processor
- 64-bit Windows operating system

Attention: V 9.0.0 The 32-bit version of IBM MQ Explorer is no longer supported.

For further information about Windows requirements, see IBM MQ Explorer Requirements and the following web pages:

- Windows 7 system requirements
- Windows 8 system requirements

Installation directories used for Windows operating systems

The 64-bit IBM MQ server or client, by default, installs its program directories into the 64-bit installation location: `C:\Program Files\IBM\MQ`.

Attention: ▶ V 9.0.0 The 32-bit client version of IBM MQ is no longer supported.

The default data directory that is used by IBM MQ changed in Version 8.0 to `C:\ProgramData\IBM\MQ`. This change affects both servers, in 32 and 64 bits and clients in 64 bits. However, if there has been a previous installation of IBM MQ on the machine on which you are installing, the new installation continues to use the existing data directory location. For more information, see Program and data directory locations.

Installing prerequisite software

To install the prerequisite software that is provided on the IBM MQ Server DVD (which does not include service packs or web browsers), choose one of the following options:

- Use the IBM MQ installation procedure.

  When you install using the IBM MQ Server DVD, there is a **Software Prerequisites** option in the IBM MQ Installation Launchpad window. You can use this option to check what prerequisite software is already installed and what is missing, and then install any missing software.
- Use Windows Explorer:
  1. Use Windows Explorer to select the `Prereqs` folder on the IBM MQ Server DVD.
  2. Select the folder for the software item to be installed.
  3. Start the installation program.

Related concepts:

"Hardware and software requirements on Linux systems" on page 331
Before you install IBM MQ , check that your system meets the hardware and operating system software requirements for the particular components you intend to install.

▶ IBM i "Hardware and software requirements on IBM i systems" on page 299
Check that the server environment meets the prerequisites for installing IBM MQ for IBM i. Check the product readme files and install missing prerequisite software supplied on the server CD.

Related tasks:

"Checking requirements on Windows" on page 444
Before you install IBM MQ on Windows, you must check for the latest information and system requirements.

Related information:

IBM MQ Explorer Requirements

Planning to install IBM MQ on Windows

▶ Windows

Before you install IBM MQ on Windows, you must choose which components to install and where to install them. You must also make some platform-specific choices.

About this task

The following steps provide links to additional information to help you with planning your installation of IBM MQ on Windows.

As part of your planning activities, make sure that you review the information on hardware and software requirements for the platform on which you are planning to install IBM MQ. For more information, see "Checking requirements on Windows" on page 444.

Procedure

1. Decide which IBM MQ components and features to install. See "IBM MQ components and features" on page 192.

   Important: Ensure that your enterprise has the correct license, or licenses, for the components that you are going to install. For more information, see "License requirements" on page 194 and IBM MQ license information.
2. Review the options for naming your installation. In some cases, you can choose an installation name to use instead of the default name. See "Installation name on UNIX, Linux, and Windows" on page 197.
3. Review the options and restrictions for choosing an installation location for IBM MQ. For more information, see "Installation location on Multiplatforms" on page 198.
4. If you plan to install multiple copies of IBM MQ, see "Multiple installations on UNIX, Linux, and Windows" on page 200.
5. If you already have a primary installation, or plan to have one, see "Primary installation on UNIX, Linux, and Windows" on page 202.
6. Make sure that the communications protocol needed for server-to-server verification is installed and configured on both systems that you plan to use. For more information, see "Server-to-server links on UNIX, Linux, and Windows" on page 210.

Additional Windows features prerequisite checking

Windows V 9.0.2

There are two Windows installation features that have additional prerequisite checking enabled in the Windows IBM MQ installer from IBM MQ Version 9.0.2. These are the `Server` feature and the `Java Runtime Environment` (JRE) feature. These features are required by other features and installing those features, without these prerequisite checks, would cause those features to be unusable.

If you perform a Graphical User Interface installation, and select the **custom install** option, you can deselect the `JRE` or `Server` features.

Attention: Dialog panels prevent you from completing the installation, until you have resolved any issues.

If you perform a silent installation, and you elect to **`REMOVE`** the `Server` or `JRE` features while installing any other features that require those features, the `Server` and `JRE` features, as appropriate, will be added to your selected installation features.

Table Table 60 on page 448 describes how the selection of certain installation features requires the `Server` or `JRE` to be added automatically.

Table 60. Installation features requiring either the Server or JRE feature

	Required by	Non-interactive name
Server	Web Administration	Web
JRE	IBM MQ Explorer Telemetry Service Managed File Transfer Service Managed File Transfer Agent Managed File Transfer Logger Managed File Transfer Tools AMQP Service Web Administration	Explorer XR Service MFT Service MFT Agent MFT Logger MFT Tools AMQP Service Web

To check whether the `JRE` or `Server` features have been installed, look in the `[INSTALLDIR]\swidtag` directory. If the:
- `ibm.com_IBM_MQ-9.0.`**`x`**`.swidtag` file is present, the `Server` has been installed
- `IBM_MQ_JRE-1.8.0.mqtag` file is present, the `JRE` has been installed.

If this is not what you require, consult the installation log.

Important: Each of the `JRE` and `Server` features are part of the set of Windows standard IBM MQ installation features. To remove the `JRE` (or the `Server`) when installing silently, add the feature to the **`REMOVE`** directive, do not merely omit it from the **`ADDLOCAL`** directive. See "Windows standard installation features" on page 443 for further details.

Installation methods for Windows

▶ Windows

If you install IBM MQ on Windows, there are several different installation types to choose from.

If you are migrating from an earlier version of IBM MQ, see Migration planning before moving to the latest version of IBM MQ. To modify an existing installation, see "Modifying a server installation" on page 469.

Interactive or Non-Interactive installation

IBM MQ for Windows is installed using the Microsoft Installer (MSI). You can use the Installation Launchpad to invoke MSI, this process is called an attended or interactive installation. Or, you can invoke MSI directly for a silent installation, without using the IBM MQ Installation Launchpad. This means that you can install IBM MQ on a system without interaction. This process is called unattended, silent, or non-interactive installation, and is useful for installing IBM MQ over a network on a remote system.

For a list of interactive and non-interactive features, see "IBM MQ features for Windows systems" on page 436.

Interactive installation

If you choose an interactive installation, before you install, you must decide what type of installation you require. Table 61 shows the installation types available, and the features that are installed with each option. For the prerequisites required for each feature, see System Requirements for IBM MQ.

The installation types are:
- Typical installation
- Compact installation
- Custom Installation

You can also:
- Specify the installation location, name, and description.
- Have multiple installations on the same computer.

See "Primary installation on UNIX, Linux, and Windows" on page 202 for important information about these features, including whether to designate your installation as the *primary installation*.

Table 61. Features installed with each type of interactive installation

Installation type	Server Features installed	Client Features installed	Comments
Typical	• Server • IBM MQ Explorer • Development Toolkit • LTS Java and .NET Messaging and Web Services V 9.0.1 Extended Messaging APIs • V 9.0.1 Web Administration	• LTS Windows Client V 9.0.1 MQI Client • Development Toolkit • LTS Java and .NET Messaging and Web Services V 9.0.1 Extended Messaging APIs	The default option. Features are installed to default locations with a default installation name. Java and .NET Messaging and Web Services (known as Extended Messaging APIs from Version 9.0.1) includes IBM MQ classes for .NET, support for the Microsoft Windows Communication Foundation (WCF) for use with Microsoft.NET 3.
Compact	• Server only	• LTS Windows Client V 9.0.1 MQI Clientonly	The feature is installed to the default location with a default installation name.

Table 61. Features installed with each type of interactive installation (continued)

Installation type	**Server Features installed**	**Client Features installed**	**Comments**
Custom	By default, the following features are preselected: • Server • IBM MQ Explorer • Development Toolkit • LTS Java and .NET Messaging and Web Services V 9.0.1 Extended Messaging APIs • V 9.0.1 Web Administration A custom installation can also install: • Telemetry Service • Advanced Message Security • Managed File Transfer Service • Managed File Transfer Logger • Managed File Transfer Agent • Managed File Transfer Tools • LTS Windows Client V 9.0.1 MQI Client	By default, the following features are preselected: • LTS Windows Client V 9.0.1 MQI Client • Development Toolkit • LTS Java and .NET Messaging and Web Services V 9.0.1 Extended Messaging APIs	A server custom installation can be used if you want to install the Windows client from within the server image. All the available features are listed and you can select which ones to install, and where to install them. You can also name and provide a description for the installation. Use a custom installation when you want to specify that the installation is primary. Java and .NET Messaging and Web Services (known as Extended Messaging APIs from Version 9.0.1) includes IBM MQ classes for .NET, support for the Microsoft Windows Communication Foundation (WCF) for use with Microsoft.NET 3 or later.

If Microsoft.NET is not installed before IBM MQ and you add it, rerun **`setmqinst -i -n Installationname`** if this is a primary installation.

The following table describes which level of .NET is required for which function:

Table 62. Required levels of Microsoft.NET

IBM MQ function	**.NET version required**
IBM MQ classes for .NET. For more information, see: Getting started with IBM MQ classes for .NET 2	.NET 2
The IBM MQ custom channel for WCF. For more information, see Developing WCF applications with IBM MQ. To build the sample solution files, either the Microsoft.NET 3.5 SDK, or Microsoft Visual Studio 2008 is needed. For more information, see: Software requirements for the WCF custom channel for IBM MQ	.NET framework 3.5 or later

For instructions on how to install IBM MQ on Windows systems, see Installing IBM MQ Server on Windows systems and “Installing an IBM MQ client on Windows” on page 481.

Non-interactive installation

If you choose a non-interactive installation the system on which you want to install must be able to access the IBM MQ image, or a copy of the files, and you must be able to access the system.

If you are running IBM WebSphere MQ Version 7.5 or later, with User Account Control (UAC) enabled, you must invoke the non-interactive installation from an elevated command prompt. Elevate a command prompt by using a right-click to start the command prompt and choose **Run as administrator**. If you try to silently install from a non-elevated command prompt, the installation fails with an error of AMQ4353 in the installation log.

There are several ways to invoke MSI:
- Using the `msiexec` command with command-line parameters.
- Using the `msiexec` command with a parameter that specifies a response file. The response file contains the parameters that you normally supply during an interactive installation. See "Installing the server using msiexec" on page 455.
- Use the `MQParms` command with command-line parameters, a parameter file, or both. The parameter file can contain many more parameters than a response file. See "Installing the server using the MQParms command" on page 463.

If the system belongs to a Windows domain you may need a special domain ID for the IBM MQ service, see "Considerations when installing IBM MQ server on Windows" on page 452 for more information.

Clearing IBM MQ installation settings

When you install IBM MQ on Windows, various values, such as the location of the data directory for IBM MQ, are stored in the registry.

In addition, the data directory contains configuration files that are read at installation time. To provide a trouble free re-installation experience, these values and files persist even after the last IBM MQ installation has been removed from the machine.

This is designed to assist you, and
- Allows you to easily uninstall and reinstall
- Ensures that you do not lose any previously defined queue managers in the process.

However in some cases this feature can be an annoyance. For example, if you want to:
- Move the data directory
- Pick up the default data directory for IBM MQ Version 9.0. See Windows: changes from IBM MQ Version 8.0 for further information.
- Install as if installing on a new machine, for example, for test purposes.
- Remove IBM MQ permanently.

To assist you in these situations, IBM MQ Version 8.0 onwards supplies a Windows command file, on the root directory of the installation media, called **`ResetMQ.cmd`**.

To run the command, enter the following:

```
ResetMQ.cmd [LOSEDATA] [NOPROMPT]
```

Attention: The parameters **`LOSEDATA`** and **`NOPROMPT`** are optional. If you supply either, or both, of these parameters, the following action results:

`LOSEDATA`
Existing queue managers become unusable. However, the data remains on disk.

NOPROMPT
Configuration information is permanently removed without further prompting.

You can run this command only after the last IBM MQ installation has been removed.

Important: You should use this script with caution. The command, even without specifying the optional parameter **LOSEDATA**, can irrecoverably remove queue manager configuration.

Related concepts:

"Considerations when installing IBM MQ server on Windows"
There are some considerations relating to security that you should take into account when installing an IBM MQ server on Windows. There are some additional considerations relating to the object naming rules and logging.

Considerations when installing IBM MQ server on Windows

▶ Windows

There are some considerations relating to security that you should take into account when installing an IBM MQ server on Windows. There are some additional considerations relating to the object naming rules and logging.

Security considerations when installing IBM MQ server on a Windows system

- If you are installing IBM MQ on a Windows domain network running Active Directory Server, you probably need to obtain a special domain account from your domain administrator. For further information, and the details that the domain administrator needs to set up this special account, see Configuring IBM MQ accounts.
- When you are installing IBM MQ server on a Windows system you must have local administrator authority .
- In order to administer any queue manager on that system, or to run any of the IBM MQ control commands your user ID must belong to the *local* `mqm` or `Administrators` group . If the local `mqm` group does not exist on the local system, it is created automatically when IBM MQ is installed. A user ID can either belong to the local `mqm` group directly, or belong indirectly through the inclusion of global groups in the local `mqm` group.
- Windows versions with a User Account Control (UAC) feature restricts the actions users can perform on certain operating system facilities, even if they are members of the Administrators group. If your user ID is in the Administrators group but not the mqm group you must use an elevated command prompt to issue IBM MQ admin commands such as crtmqm, otherwise the error AMQ7077 is generated. To open an elevated command prompt, right-click the start menu item, or icon, for the command prompt, and select **Run as administrator**
- Some commands can be run without being a member of the mqm group (see Authority to administer IBM MQ).
- If you intend to administer queue managers on a remote system, your user ID must be authorized on the target system.
- As with other versions of Windows, the object authority manager (OAM) gives members of the Administrators group the authority to access all IBM MQ objects even when UAC is enabled.

Naming considerations

Windows has some rules regarding the naming of objects created and used by IBM MQ. These naming considerations apply to IBM WebSphere MQ Version 7.5 or later.

- Ensure that the machine name does not contain any spaces. IBM MQ does not support machine names that include spaces. If you install IBM MQ on such a machine, you cannot create any queue managers.
- For IBM MQ authorizations, names of user IDs and groups must be no longer than 64 characters (spaces are not allowed).

- An IBM MQ for Windows server does not support the connection of a Windows client if the client is running under a user ID that contains the @ character, for example, abc@d. Similarly, the client user ID should not be the same as local group.
- A user account that is used to run the IBM MQ Windows service is set up by default during the installation process; the default user ID is MUSR_MQADMIN. This account is reserved for use by IBM MQ. Refer to Configuring IBM MQ accounts.
- When an IBM MQ client connects to a queue manager on the server, the username under which the client runs must not be same as the domain or machine name. If the user has the same name as the domain or machine, the connection fails with return code 2035(MQRC_NOT_AUTHORIZED).

Logging

You can set up logging during installation which assists you in troubleshooting any problems you might have with the installation.

From Version 7.5, logging is enabled by default from the Launchpad. You can also enable complete logging, for more information, see How to enable Windows Installer logging.

Digital signatures

The IBM MQ programs and installation image are digitally signed on Windows to confirm that they are genuine and unmodified. From IBM MQ Version 8.0 the SHA-256 with RSA algorithm is used to sign the IBM MQ product.

Installing IBM MQ server on Windows

Windows

This topic describes how to install IBM MQ server on Windows systems either by using the Launchpad, or by using the MSI technology.

About this task

To install IBM MQ server on Windows systems, you can choose either to install with the Launchpad or to install using MSI technology. MSI provides both an interactive installation and a non interactive installation.

Procedure

- To install IBM MQ server by using the Launchpad, see “Installing the server using the Launchpad” on page 454.
- To install IBM MQ server on by using the MSI technology, see “Installing the server using msiexec” on page 455.

Related concepts:

"Modifying a server installation" on page 469
You can modify an IBM MQ server installation interactively using the launchpad or non-interactively using msiexec.

"Configuring an IBM MQ server" on page 470
After installing IBM MQ server, it is necessary to configure it.

Related tasks:

"Uninstalling IBM MQ on Windows" on page 514
You can uninstall the IBM MQ MQI clients and servers on Windows systems by using the control panel, the command line (`msiexec`), `MQParms`, or by using the installation media, in which case you can optionally remove queue managers as well.

Installing the server using the Launchpad

Windows

This topic describes how to install IBM MQ server on Windows systems by using the Launchpad. This procedure can be used for installing a first or a subsequent installation.

About this task

These instructions cover how to display the installation Launchpad window. You can use the launchpad to make a compact, typical, or custom installation of IBM MQ. You can reuse the launchpad multiple times to install further installations. It automatically selects the next available installation name, instance, and location to use. To view all the installation types and the features that are installed with each option, see "Installation methods for Windows" on page 448.

Note that if you have previously uninstalled IBM MQ from your system (see "Uninstalling IBM MQ on Windows" on page 514), some configuration information might remain, and some default values might be changed.

Procedure

1. Access the IBM MQ installation image. The location might be the mount point of the DVD, a network location, or a local file system directory.
2. Locate `setup.exe` in the base directory of the IBM MQ installation image.
   - From a DVD, this location might be `E:\setup.exe`
   - From a network location, this location might be `m:\instmqs\setup.exe`
   - From a local file system directory, this location might be `C:\instmqs\setup.exe`
3. Double-click the **Setup** icon to start the installation process. It is possible to run either by:
   - Running `setup.exe` from the command prompt. Or
   - Double-clicking `setup.exe` from Windows Explorer.

   If you are installing on a Windows system with UAC enabled, accept the Windows prompt to allow the launchpad to run as elevated. During installation, you might also see Open File - Security Warning dialog boxes that list International Business Machines Limited as the publisher. Click **Run** to allow the installation to continue.

   The IBM MQ Installation Launchpad window is displayed.
4. Continue to follow the Launchpad instructions as shown on screen.

What to do next

- If you have chosen this installation to be the primary installation on the system, you must now set it as the primary installation. Enter the following command at the command prompt:

```
MQ_INSTALLATION_PATH\bin\setmqinst -i -p MQ_INSTALLATION_PATH
```

You can have only one primary installation on a system. If there is already a primary installation on the system, you must unset it before you can set another installation as the primary installation. For more information, see Changing the primary installation.

- You might want to set up the environment to work with this installation. You can use the **setmqenv** or **crtmqenv** command to set various environment variables for a particular installation of IBM MQ. For more information, see setmqenv and crtmqenv.
- For instructions on how to verify your installation, see "Verifying an IBM MQ installation on Windows" on page 496.

Related concepts:

"Modifying a server installation" on page 469
You can modify an IBM MQ server installation interactively using the launchpad or non-interactively using msiexec.

"Configuring an IBM MQ server" on page 470
After installing IBM MQ server, it is necessary to configure it.

Related tasks:

"Installing the server using msiexec"
IBM MQ on Windows uses the MSI technology to install software. MSI provides both an interactive installation and a non interactive installation.

"Uninstalling IBM MQ on Windows" on page 514
You can uninstall the IBM MQ MQI clients and servers on Windows systems by using the control panel, the command line (**msiexec**), **MQParms**, or by using the installation media, in which case you can optionally remove queue managers as well.

Installing the server using msiexec

Windows

IBM MQ on Windows uses the MSI technology to install software. MSI provides both an interactive installation and a non interactive installation.

Before you begin

If you are running IBM MQ on Windows systems with User Account Control (UAC) enabled, you must invoke the installation with elevated privileges. If you are using the Command prompt or IBM MQ Explorer elevate privileges by using a right-click to start the program and selecting Run as administrator. If you try to run **msiexec** without using elevated privileges, the installation fails with an error of AMQ4353 in the installation log.

About this task

IBM MQ on Windows uses the MSI technology to install software. MSI provides both an interactive installation and a non interactive installation. An interactive installation displays panels and ask questions.

The **msiexec** command uses parameters to give MSI some or all of the information that can also be specified through panels during an interactive installation. This means that a user can create a reusable automated or semi-automated installation configuration. Parameters can be given through the command line, a transform file, a response file, or a combination of the three.

Procedure

To install using msiexec, at the command line, enter the **msiexec** command in the following format:

```
msiexec parameters [USEINI="response-file"] [TRANSFORMS="transform_file"]
```

Where:

parameters
: are either command-line parameters preceded by a `/` character, or property=value pairs (if using both forms of parameter always put the command-line parameters first). For further information, see "Specifying command line parameters with msiexec," which contains a link to the web site that lists all the command line parameters that are available.

  For an unattended installation, you must include the /q or /qn parameter in the command line. Without this parameter, the installation is interactive.

  Note: You must include the **`/i`** parameter and the file location of the IBM MQ installer package.

response-file
: is the full path and file name of the file that contains the [Response] stanza and the required property=value pairs, for example `C:\MyResponseFile.ini`. An example response file, `Response.ini`, is supplied with IBM MQ. This file contains default installation parameters. For further information, see "Using a response file with msiexec" on page 457.

transform_file
: is the full path and file name of a transform file. For further information, see "Using transforms with msiexec for server installation" on page 462 and "Choosing MSI Instance IDs for multiple server installations" on page 461.

Note: For a silent installation to succeed, the AGREETOLICENSE="yes" property must be defined either on the command line or in the response file.

Results

After the command has been entered, the command prompt immediately reappears. IBM MQ is installing as a background process. If you have entered parameters to produce a log, check this file to see how the installation is progressing. If the installation completes successfully, you see the message `Installation operation completed successfully` in the log file.

Specifying command line parameters with msiexec:
About this task

The **`msiexec`** command can accept two types of parameters on the command line, as follows:

- Standard command line parameters, preceded by a `/` character.

  For a table of the **`msiexec`** command line parameters, see the MSDN Command-Line Options web page.
- Property=value pair parameters on the command line. All the parameters available for use in a response file can be used on the command line, for a list of these, see Table 64 on page 458. In addition there are some extra property=value pair parameters that are only for use on the command line, for details see Table 63 on page 457.

  When using the property=value pair parameters note that:
  - Property strings must be in uppercase.
  - Value strings are not case-sensitive, except for feature names. You can enclose value strings in double quotation marks. If a value string includes a blank, enclose the blank value string in double quotation marks.
  - For a property that can take more than one value, use the format:

    ```
    ADDLOCAL="Server,Client"
    ```
  - For properties taking paths and filenames, for example PGMFOLDER, you must supply the paths as absolute paths and not relative; that is, `C:\folder\file` and not `".\folder\file"`.

  When using property=value pair and command line parameters with the **`msiexec`** command, enter command line parameters first.

If a parameter is specified both on the command line and in a response file, the setting on the command line takes precedence.

Example

Here is an example of a typical **msiexec** command. All parameters, separated by one or more spaces, must be typed on the same line as the **msiexec** call.

```
msiexec
/i "path\MSI\IBM MQ.msi"
/l*v c:\install.log
/q
TRANSFORMS="1033.mst"
AGREETOLICENSE="yes"
ADDLOCAL="Server"
```

Here is an example of a typical **msiexec** command when you are installing a second copy of IBM WebSphere MQ Version 7.5, or later. All parameters, separated by one or more spaces, must be typed on the same line as the **msiexec** call.

```
msiexec
/i "path\MSI\IBM MQ.msi"
/l*v c:\install.log
/q
TRANSFORMS=":InstanceId2.mst;1033.mst"
AGREETOLICENSE="yes"
ADDLOCAL="Server"
MSINEWINSTANCE=1
```

Where `/l*v c:\install.log` writes installation log to file `c:\install.log`.

The following table shows the parameters which can only be provided on the command line and not in a response file.

Table 63. msiexec property=value parameters

Property	Values	Meaning
USEINI	*path* \ *file_name*	Use the specified response file. See "Using a response file with msiexec"
SAVEINI	*path* \ *file_name*	Generate a response file during installation. The file contains those parameters selected for this installation that a user might make during an interactive installation.
ONLYINI	1|yes| ""	1, yes or any value other than null. End the installation before updating the target system, but after generating a response file, if this is specified. "". Continue the installation and update the target system (the default).
TRANSFORMS	:InstanceId *x*.mst| *path* \ *file_name* | :InstanceId *x*.mst; *path* \ *file_name*	The :InstanceId *x*.mst value is only required for a subsequent installation of IBM MQ. The *path* \ *file_name* specifies what transform (.mst) files must be applied to the product. For example, `"1033.mst"` specifies the supplied U.S. English transform file.
MSINEWINSTANCE	1	This property is only required for subsequent installations of IBM MQ

Using a response file with msiexec:

About this task

You can use the **msiexec** command with a parameter which specifies additional properties that are defined in a response file. You can combine the msiexec command-line parameters described in “Specifying command line parameters with msiexec” on page 456.

A response file is an ASCII text file, with a format like a Windows .ini file, that contains the stanza [Response]. The [Response] stanza contains some or all the parameters that would normally be specified as part of an interactive installation. The parameters are given in a property=value pair format. Any other stanzas in the response file are ignored by **msiexec**. An example response file, Response.ini, is supplied with IBM MQ. It contains the default installation parameters.

Procedure

A typical example of an msiexec command is: `msiexec /i "path\MSI\IBM MQ.msi" /l*v c:\install.log TRANSFORMS= "1033.mst" USEINI= "C:\MQ\Responsefile"`
If a parameter is specified both on the command line and in a response file, the setting on the command line takes precedence. All the parameters available for use in a response file can also be used on the command line, for a list of these see Table 64.
In the response file, all text is in English, and comments begin with a **;** character.
For information about creating a response file, see “Creating a response file for server installation” on page 463.

Example

An example of a typical response file:

```
[Response]
PGMFOLDER="c:\mqm"
DATFOLDER="c:\mqm\data"
LOGFOLDER="c:\mqm\log"
AGREETOLICENSE="yes"
LAUNCHWIZ=""
WIZPARMFILE="d:\MQParms.ini"
ADDLOCAL="Server,Client"
REMOVE="Toolkit"
```

Table 64. Response file parameters

Property	Values	Meaning
PGMFOLDER	*path*	Folder for the IBM MQ program files. For example, c:\mqm.
DATFOLDER	*path*	Folder for the IBM MQ data files. For example, c:\mqm\data. **Note:** Multiple installations of IBM MQ all use the same **DATFOLDER**.
LOGFOLDER	*path*	Folder for the IBM MQ queue manager log files. For example, c:\mqm\log. **Note:** Multiple installations of IBM MQ all use the same **LOGFOLDER**.

Table 64. Response file parameters (continued)

Property	**Values**	**Meaning**
USERCHOICE	0|no	If the command line or response file specifies parameters to install features, a dialog can be displayed to prompt the user to accept the preselected options, or review and possibly change them. 0 or no. Suppresses display of the dialog. Anything else. Dialog is displayed. Not used for a silent installation.
AGREETOLICENSE	yes	Accept the terms of the license. Set to `yes` before a silent installation. If the installation is not silent, this parameter is ignored.
KEEPQMDATA	**keep** |delete	If the Server feature is to be uninstalled, whether to delete any existing queue managers. delete removes any existing queue managers. keep, or any other value, keeps them. **Note:** This property is only valid on a final server uninstallation. Otherwise this property is ignored.
LAUNCHWIZ	0|1|yes|no| **""**	0 or no. Do not launch the Prepare IBM MQ wizard after IBM MQ is installed. 1 or yes. Launch the Prepare IBM MQ wizard if the Server feature is installed. "". Launch the Prepare IBM MQ wizard to install the Server (the default). If this option is to launch the Prepare IBM MQ wizard, you can specify the WIZPARMFILE, either in this file, or on the command line. The Prepare IBM MQ wizard must be run to make your IBM MQ installation operational. If you choose not to launch it here, you must run it before using IBM MQ.
WIZPARMFILE	*path* \ *file_name*	When specified, the file that contains the parameters to pass to the Prepare IBM MQ wizard when it is launched. These are in the [Services].
ADDLOCAL	*feature, feature, All* | ””	A comma-separated list of features to install locally. For a list of valid feature names, see “IBM MQ features for Windows systems” on page 436. All installs all features "" installs the typical features. If you do not want a feature use `REMOVE="`*`feature`*`"` **Note:** If this is a new installation the typical features are installed by default irrespective of the feature list provided in the ADDLOCAL property. If you do not want a feature use `REMOVE="`*`feature`*`"`

Table 64. Response file parameters (continued)

Property	Values	Meaning
REMOVE	*feature*, *feature*, |All| ""	A comma-separated list of features to remove. For a list of valid feature names, see "IBM MQ features for Windows systems" on page 436. All uninstalls all features "" uninstalls no features (the default).
STARTSERVICE	0|no| ""	0 or no. Do not start the IBM MQ Service at the end of installation. "" (the default). Start the IBM MQ Service at the end of installation if it was running at the start, or if this is a new installation. Anything else. Start the Service at the end of the installation. Ignored if the server feature is not installed. If you do not start the IBM MQ Service, IBM MQ will not be operational and queue managers will not start. You must run the Prepare IBM MQ wizard for the service to be correctly configured. This parameter is only valid if LAUNCHWIZ is set to no.
STARTTASKBAR	0|no| ""	0 or no. Do not start the IBM MQ taskbar application at the end of installation. "" (the default). Start the IBM MQ taskbar application at the end of installation if it was running at the start, or if this is a new installation. Anything else. Start the taskbar application at the end of the installation. Ignored if the server feature is not installed. This parameter is only valid if LAUNCHWIZ is set to no.
INSTALLATIONDESC	"Description of installation"	Sets the installation description from the command line. Subject to the documented installation description length limitations
INSTALLATIONNAME	[INSTALLATION0,]Name	Sets the installation name from the command line. Subject to the documented installation name character and length limitations. **Note:** Supply INSTALLATION0,Name only when upgrading from versions of the product before IBM WebSphere MQ Version 7.1.
MAKEPRIMARY	0|1| ""	Makes the installation primary, if possible, or removes the primary flag. 1 = Make primary, 0 = Make non-primary, - use default algorithm **Note:** This option is ignored if a version of the product before IBM WebSphere MQ Version 7.1 is installed, or if another installation of IBM WebSphere MQ Version 7.1, or later, is present and set as the primary.

The typical features include the following features:
- Server
- MQ Explorer
- Java and .NET Messaging and Web Services V 9.0.1 Renamed to Extended Messaging APIs from Version 9.0.1
- V 9.0.1 Web Administration
- Development Toolkit
- V 9.0.2 Java Runtime Environment

Related tasks:

"Choosing MSI Instance IDs for multiple server installations"
For multiple silent installations, for each version that is installed you must find an MSI instance ID that is available to use for that installation.

"Creating a response file for server installation" on page 463
A response file is used with **`msiexec`**. You can create it in three ways.

"Installing the server using the MQParms command" on page 463
You can use the **`MQParms`** command to invoke installation or uninstallation of the IBM MQ server.

Related reference:

"Using transforms with msiexec for server installation" on page 462

Choosing MSI Instance IDs for multiple server installations: Windows

For multiple silent installations, for each version that is installed you must find an MSI instance ID that is available to use for that installation.

About this task

In order to support silent, or non-interactive, multiple installations, you need to find out whether the instance ID you want to use is already in use or not and choose the appropriate one. For each installation media (for example, each client and server), Instance ID 1 is the default ID which is used for single installations. If you want to install alongside Instance ID 1 you need to specify which instance you want to use. If you have already installed instance 1, 2, and 3 then you need to find out what the next available instance is, for instance, Instance ID 4. Similarly, if instance 2 has been removed, you need to find out that there is a gap that can be reused. You can find out which Instance ID is currently in use by using the **`dspmqinst`** command.

Procedure

1. Type **`dspmqinst`** to find a free MSI Instance in the media being installed by reviewing the MSIMedia and MSIInstanceId values for the versions already installed. For example:
   ```
   InstName: Installation1
   InstDesc:
   Identifier:    1
   InstPath:      C:\Program Files\IBM\MQ
   Version:       9.0.0.0
   Primary:       Yes
   State:         Available
   MSIProdCode:   {74F6B169-7CE6-4EFB-8A03-2AA7B2DBB57C}
   MSIMedia:      9.0 Server
   MSIInstanceId: 1
   ```
2. If MSI Instance ID 1 is in use and you want to use MSI Instance ID 2, the following parameters must be added to the msiexec call:
   ```
   MSINEWINSTANCE=1 TRANSFORMS=":instanceId7.mst;1033.mst"
   ```

What to do next

For multiple installations, the **INSTALLATIONNAME** or **PGMFOLDER** must be supplied as an additional parameter on any non-interactive installation command. Supplying the **INSTALLATIONNAME** or **PGMFOLDER** ensures that you do not work with the wrong installation in case you omit or incorrectly specify the **TRANSFORMS** parameter.

Using transforms with msiexec for server installation: Windows

MSI can use transforms to modify an installation. During IBM MQ installation, transforms can be used to support different national languages. IBM MQ is supplied with transform files in the `\MSI` folder of the Server image. These files are also embedded in the IBM MQ Windows installer package, `IBM MQ.msi`.

On the **msiexec** command line, you can specify the required language by using the TRANSFORMS property in a property=value pair. For example:

```
TRANSFORMS="1033.mst"
```

You can also specify the full path and file name of the transform file. Again, the quotation marks surrounding the value are optional. For example:

```
TRANSFORMS="D:\Msi\1033.mst"
```

Table 65 shows the locale identifier, language, and the transform file name to use in the **msiexec** command line.

You might need to merge transforms to install multiple installations of the same version, for example:

```
TRANSFORMS=":InstanceId2.mst;D:\Msi\1033.mst"
```

You can also specify the required language by using the MQLANGUAGE property with the **MQParms** command. For information about the msiexec property=value parameters, see “MQParms parameter file - server installation” on page 465.

Parameters

Table 65. Supplied transform files for various language support. This table shows the supplied transform files, the resulting language, and the numeric value to use in the **msiexec** command line.

Language	Transform File name	Value
U.S. English	1033.mst	1033
German	1031.mst	1031
French	1036.mst	1036
Spanish	1034.mst	1034
Italian	1040.mst	1040
Brazilian Portuguese	1046.mst	1046
Japanese	1041.mst	1041
Korean	1042.mst	1042
Simplified Chinese	2052.mst	2052
Traditional Chinese	1028.mst	1028
Czech	1029.mst	1029
Russian	1049.mst	1049
Hungarian	1038.mst	1038
Polish	1045.mst	1045

Creating a response file for server installation: Windows

A response file is used with **msiexec**. You can create it in three ways.

About this task

A response file is used with the **msiexec** command. For further information, see "Using a response file with msiexec" on page 457.

Procedure

There are three ways to create a response file for installation:
- Copy and edit the file `Response.ini` that is supplied on the IBM MQ Windows Server DVD, using an ASCII file editor.
- Create your own response file using an ASCII file editor.
- Use the **msiexec** command with the **SAVEINI** (and optionally, the **ONLYINI**) command line parameters to generate a response file that contains the same installation options. See Table 63 on page 457.

Example

A typical example of using **msiexec** with the **SAVEINI** parameter is here:

```
msiexec /i "path\IBM MQ.msi" /q SAVEINI="response_file"
TRANSFORMS="1033.mst" AGREETOLICENSE="yes"
```

Installing the server using the MQParms command: Windows

You can use the **MQParms** command to invoke installation or uninstallation of the IBM MQ server.

Before you begin

The **MQParms** command can use parameters on a command line, or those specified in a parameter file. The parameter file is an ASCII text file that contains the parameter values that you want to set for the installation. The **MQParms** command takes the specified parameters and generates the corresponding **msiexec** command line.

This means that you can save all the parameters that you want to use with the **msiexec** command in a single file.

If you are running IBM MQ on Windows systems with User Account Control (UAC) enabled, you must invoke the installation with elevated privileges. If you are using the Command prompt or IBM MQ Explorer elevate privileges by using a right-click to start the program and selecting **Run as administrator**. If you try to run the `MQParms` program without using elevated privileges, the installation fails with an error of AMQ4353 in the installation log.

For silent operations, this must include the **/q** or **/qn** parameter, either on the command line, or in the [MSI] stanza of the parameter file. You must also set the AGREETOLICENSE parameter to "yes".

You can specify many more parameters in the parameter file that you use with the `MQParms` command than you can in the response file that you use directly with the **msiexec** command. Also, as well as parameters that the IBM MQ installation uses, you can specify parameters that can be used by the Prepare IBM MQ wizard.

If you do not complete the Prepare IBM MQ Wizard directly after IBM MQ installations or if for any reason your machine is rebooted between completing IBM MQ installation and completing the Prepare

IBM MQ Wizard, ensure that the wizard is run with Administrator privilege afterward, otherwise the installation is incomplete, and might fail. You might also see Open File - Security Warning dialog boxes that list International Business Machines Limited as the publisher. Click **Run** to allow the wizard to continue

An example of the file `MQParms.ini` is supplied with IBM MQ. This file contains default installation parameters.

There are two ways to create a parameter file for installation:
- Copy and edit the file `MQParms.ini` that is supplied with the product, using an ASCII file editor.
- Create your own parameter file using an ASCII file editor.

About this task

To invoke installation using the **`MQParms`** command:

Procedure

1. From a command line, change to the root folder of the IBM MQ Server DVD (that is, the location of the file MQParms.exe).
2. Enter the following command:
   ```
   MQParms parameter_file parameters ]
   ```
   where:

   parameter_file
      is the file that contains the required parameter values. If this file is not in the same folder as MQParms.exe, specify the full path and file name. If you do not specify a parameter file, the default is `MQParms.ini`. For silent installation, the `MQParms_silent.ini` parameter file can be used. For further details, see "MQParms parameter file - server installation" on page 465.

   parameters
      are one or more command-line parameters, for a list of these, see the MSDN Command-Line Options web page.

Example

A typical example of an **`MQParms`** command is:
```
MQParms "c:\MyParamsFile.ini" /l*v c:\install.log
```

A typical example of an **`MQParms`** command when you are installing a second copy of IBM MQ is:
```
MQParms "c:\MyParamsFile.ini" /l*v c:\install.log TRANSFORMS=":InstanceId2.mst;1033.mst" MSINEWINSTANCE=1
```

Alternatively, TRANSFORMS and MSINEWINSTANCE can be specified in the MSI stanza of the parameter file.

If you specify a parameter both on the command line and in the parameter file, the setting on the command line takes precedence.

If you specify a parameter file, you might want to run the encryption utility before you use the **`MQParms`** command (see "Encrypting a parameter file" on page 468).

If you do not specify `/i`, `/x`, `/a`, or `/j`, **`MQParms`** defaults to standard installation using the IBM MQ Windows Installer package, IBM MQ.msi. That is, it generates the following part of the command line:
```
/i " current_folder \MSI\IBM MQ.msi"
```

If you do not specify a WIZPARMFILE parameter, **MQParms** defaults to the current parameter file. That is, it generates the following part of the command:

```
WIZPARMFILE=" current_folder \ current_parameter_file "
```

MQParms parameter file - server installation: Windows

A parameter file is an ASCII text file that contains sections (stanzas) with parameters that can be used by the **MQParms** command. Typically, this is an initialization file such as `MQParms.ini`.

The **MQParms** command takes parameters from the following stanzas in the file:

[MSI] Contains general properties related to how the **MQParms** command runs and to the installation of IBM MQ.

The properties that you can set in this stanza are listed in “Installing the server using msiexec” on page 455, and Table 66 on page 466.

[Services]
Contains properties related to IBM MQ account configuration, in particular, the user account required for IBM MQ Services. If you are installing IBM MQ on a network where the domain controller is on a Windows 2003 server, you probably need details of a special domain account. For further information, see “Configuring IBM MQ accounts” on page 475 and “Configuring IBM MQ with the Prepare IBM MQ wizard” on page 471.

The properties that you can set in this stanza are listed in Table 68 on page 467.

MQParms ignores any other stanzas in the file.

The stanza parameters are in the form property=value, where property is always interpreted as uppercase, but value is case sensitive. If a value string includes a blank, it must be enclosed in double quotation marks. Most other values can be enclosed in double quotation marks. Some properties can take more than one value, for example:

```
ADDLOCAL="Server,Client"
```

To clear a property, set its value to an empty string, for example:

```
REINSTALL=""
```

The following tables show the properties that you can set. The default is shown in bold.

For the [MSI] stanza, you can enter standard MSI command line options and properties. For example:

```
- /q
- ADDLOCAL="server"
- REBOOT=Suppress
```

Refer to Table 66 on page 466, Table 67 on page 466, and Table 68 on page 467 for the properties used to install IBM MQ.

Table 66 on page 466 shows additional properties in the stanza that affect how the `MQParms` command runs, but that do not affect the installation.

Table 66. Properties used by MQParms in the MSI stanza

Property	Values	Description
MQPLOG	*path* | *file_name*	**MQParms** generates a text log file with the specified name and location.
MQPLANGUAGE	**system** | user | *transform_value* | existing	The installation language. system. Install using the language of the default system locale (the default). user. Install using the language of the default locale of the user. *transform_value*. Install using the language specified by this value. See Table 67. existing. If IBM MQ already exists on the system, the same language will be used by default, otherwise system is used.
MQPSMS	**0** | no	0 or no. **MQParms** does not wait for the **msiexec** command to end (the default). Any other value. **MQParms** waits for the **msiexec** command to end.
MQPINUSE	**0** | 1	If MQPINUSE is set to 1, **MQParms** continues installing even if IBM MQ files are in use. If this option is used a reboot will be required to complete the installation.

Table 67. Valid values for the MQPLANGUAGE property

Language	Valid values		
U.S. English	English	en_us	1033
German	German	de_de	1031
French	French	fr_fr	1036
Spanish	Spanish	es_es	1034
Italian	Italian	it_it	1040
Brazilian Portuguese	Brazilian Portuguese	pt_br	1046
Japanese	Japanese	ja_jp	1041
Korean	Korean	ko_kr	1042
Simplified Chinese	Simplified Chinese	zh_cn	2052
Traditional Chinese	Traditional Chinese	zh_tw	1028
Czech	Czech	cs_cz	1029
Russian	Russian	ru_ru	1049
Hungarian	Hungarian	hu_hu	1038
Polish	Polish	pl_pl	1045

For the [Services] stanza, you can enter parameters in property=value format. You might want to encrypt the values in this stanza. See “Encrypting a parameter file” on page 468.

Table 68. Properties used in the Services stanza

Property	Values	Description
USERTYPE	**local** | domain | onlydomain	The type of user account to use: **local** Creates a local user account. **domain** Creates a local user account. If this does not have the required security authorities, it uses the domain user account specified by DOMAINNAME, USERNAME, and PASSWORD. **onlydomain** Does not create a local user account, but immediately uses the domain user account specified by DOMAINNAME, USERNAME and PASSWORD. If any of these three properties are missing, a USERTYPE of local is assumed. The properties DOMAINNAME, USERNAME, and PASSWORD are required if USERTYPE is set to onlydomain.
DOMAINNAME	*domain_name* 1	The domain for the domain user account. Required if USERTYPE is set to domain or onlydomain.
USERNAME	*user_name* 1	The user name for the domain user account. Required if USERTYPE is set to domain or onlydomain..
PASSWORD	*password* 1	The password for the domain user account. Required if USERTYPE is set to domain or onlydomain.
1. Do not enclose this value in double quotation marks.		

A typical example of a parameter file is:

```
[MSI]
MQPLANGUAGE=1033
MQPLOG=%temp%\MQParms.log
MQPSMS=no
ADDLOCAL=Server
/m miffile
REMOVE=""
/l*v c:\install.log

[Services]
USERTYPE=domain
DOMAINNAME=mqm*df349edfcab12
USERNAME=mqm*a087ed4b9e9c
PASSWORD=mqm*d7eba3463bd0a3
```

Encrypting a parameter file: Windows

If the DOMAINNAME, USERNAME, and PASSWORD values in the [Services] stanza of a parameter file are not already encrypted, you can encrypt them by running the `setmqipw` utility.

About this task

Use the `setmqipw` utility to encrypt the DOMAINNAME, USERNAME, and PASSWORD values in the [Services] stanza of a parameter file, if they are not already encrypted. (These values might be encrypted if you have run the utility before.) `setmqipw` will also encrypt the QMGRPASSWORD and CLIENTPASSWORD values in the [SSLMigration] stanza of a parameter file.

This encryption means that, if you need a special domain account to configure IBM MQ (see “Configuring IBM MQ accounts” on page 475), or you need to keep key database passwords secret, details are kept secure. Otherwise, these values, including the domain account password, flow across the network as clear text. You do not have to use this utility, but it is useful if security in your network is an issue.

To run the script:

Procedure

1. From a command line, change to the folder that contains your parameter file.
2. Enter the following command:
   ```
   CD_drive:\setmqipw
   ```
   Note: You can run the command from a different folder, by entering the following command, where *parameter_file* is the full path and file name of the parameter file:
   ```
   CD_drive:\setmqipw parameter_file
   ```

Results

If you view the resulting parameter file, the encrypted values start with the string `mqm*`. Do not use this prefix for any other values; passwords or names that begin with this prefix are not supported.

The utility creates a log file, `setmqipw.log`, in the current directory. This file contains messages related to the encryption process. When encryption is successful, messages are similar to:

```
Encryption complete
Configuration file closed
Processing complete
```

What to do next

After you encrypt the parameter file, you can use it in the normal way with the `MQParms` command (see “Installing the server using the MQParms command” on page 463).

Modifying a server installation

Windows

You can modify an IBM MQ server installation interactively using the launchpad or non-interactively using msiexec.

Related concepts:

"Modifying a server installation silently using msiexec" on page 470
You can silently remove or install IBM MQ features on Windows by using `msiexec`.

Related tasks:

"Modifying a server installation using the Installation Launchpad"
You can interactively remove or install IBM MQ features on Windows by using IBM MQ Installation Launchpad.

Modifying a server installation using the Installation Launchpad: Windows

You can interactively remove or install IBM MQ features on Windows by using IBM MQ Installation Launchpad.

Before you begin

To modify an installation, some features of IBM MQ must already be installed.

About this task

To remove or install IBM MQ features follow the instructions. This procedure is the only way to interactively remove or install IBM MQ features on Windows Server 2008:

Procedure

1. Insert the IBM MQ for Windows Server DVD into the DVD drive.
2. If autorun is installed, the installation process starts.

   Otherwise, double-click the **Setup** icon in the root folder of the DVD to start the installation process.

   The IBM MQ Installation Launchpad window is displayed.
3. Click the **IBM MQ Installation** option.
4. Click **Launch IBM MQ Installer**. Wait until the IBM MQ Setup window is displayed with a welcome message.
5. If you have multiple installations on your system, you must choose the installation you want to modify. Do this by selecting the **Maintain or upgrade an existing instance** option and choosing the appropriate instance. If you are upgrading a IBM WebSphere MQ Version 7.0.1 installation (or earlier) to Version 7.1.0, and you already have a Version 7.1.0 or greater installation, you need to select **Install a new instance**. A subsequent panel then allows you to choose the installation you would like to upgrade.
6. Click **Next** to continue. The Program Maintenance panel is displayed.
7. Select **Modify**, then click **Next**.

   The Features panel is displayed.
8. Click the **+** symbol next to a feature to show any dependent features (subfeatures).
9. To change the installation of a feature:
   a. Click the symbol next to the feature name to display a menu.
   b. Select the required option from:
      - Install this feature
      - Install this feature and all its subfeatures (if any)

- Do not install this feature (remove if already installed)

  The symbol next to the feature name changes to show the current installation option.
10. Stop the web server before removing the web feature. If you do not do this, you receive an error message.
11. When your selections are complete, click **Next**. IBM MQ installation begins.

What to do next

After modifying the installation, you might need to run **setmqenv** again as described in *What to do next* in "Installing IBM MQ server on Windows" on page 453.

Modifying a server installation silently using msiexec: Windows

You can silently remove or install IBM MQ features on Windows by using **msiexec**.

To silently modify an installation using **msiexec**, set the ADDLOCAL parameter to include the features you want to add, and set the REMOVE parameter to the features you want to remove.

For example if you use ADDLOCAL="JavaMsg" and REMOVE="" it modifies the installation to include the Java Messaging and Web Services feature.

```
msiexec /i {product code} /q ADDLOCAL="JavaMsg" REMOVE="" INSTALLATIONNAME="Installation1"
```

where *product_code* is the value shown for MSIProdCode in the output of the following command:

```
dspmqinst -n installation_name
```

An example of a product code is `{0730749B-080D-4A2E-B63D-85CF09AE0EF0}`.

The instructions for msiexec begin here: "Installing the server using msiexec" on page 455.

Configuring an IBM MQ server

Windows

After installing IBM MQ server, it is necessary to configure it.

The configuration described in this section is for an environment that uses TCP/IP. The configuration procedure is the same for environments that use other communications protocols (for example, SNA, SPX, or NetBIOS). However, not all of the functions and facilities of IBM MQ for Windows are available in these environments. The items that are not available are:
- IBM MQ Postcard
- IBM MQ Explorer

If you are setting up IBM MQ for use with the Microsoft Cluster Service (MSCS), see Supporting the Microsoft Cluster Service (MSCS) for more information.

Using IBM MQ remotely: Windows

To create and start queue managers when connected remotely, you must have the **`Create global objects`** user access.

If you are connecting to a Windows machine using either Terminal Services or a Remote Desktop Connection and you have problems creating, starting or deleting a queue manager this might be because of the user access **`Create global objects`**.

The **`Create global objects`** user access limits the users authorized to create objects in the global namespace. In order for an application to create a global object, it must either be running in the global namespace, or the user under which the application is running must have the **`Create global objects`** user access applied to it.

When you connect remotely to a Windows machine using either Terminal Services or Remote Desktop Connection, applications run in their own local namespace. If you attempt to create or delete a queue manager using IBM MQ Explorer or the **`crtmqm`** or **`dltmqm`** command, or to start a queue manager using the **`strmqm`** command, it results in an authorization failure. This creates an IBM MQ FDC with Probe ID XY132002.

Starting a queue manager using the IBM MQ Explorer, or using the **`amqmdain qmgr start`** command works correctly because these commands do not directly start the queue manager. Instead the commands send the request to start the queue manager to a separate process running in the global namespace.

If you need to perform any of these operations on a queue manager when connected remotely to a Windows machine, you must have the **`Create global objects`** user access. For information on how to assign a user this access, see your operating system documentation.

Administrators have the **`Create global objects`** user access by default, so if you are an administrator you can create and start queue managers when connected remotely without altering your user rights.

Configuring IBM MQ with the Prepare IBM MQ wizard: Windows

The Prepare IBM MQ wizard helps you to configure IBM MQ files and a user account for your network, and migrate any queue managers and data from a previous installation.

About this task

You must run the Prepare IBM MQ wizard to configure the IBM MQ Service before you can start any queue managers.

The Prepare IBM MQ wizard window is displayed when IBM MQ installation completes. Follow the instructions given by the wizard to configure IBM MQ. At any time while the wizard is running you can click **More Information** in the wizard to view online help about the task you are doing.

On Windows systems, you must do this task under a Windows administrator account, or domain administrator account in case your workstation is a member of a Windows domain.

On Windows systems with UAC enabled, if you do not complete the Prepare IBM MQ Wizard directly after IBM MQ is installed, or if for any reason your machine is rebooted between completing IBM MQ installation and completing the Prepare IBM MQ Wizard, you must accept the Windows prompt when it appears to allow the wizard to run as elevated.

Procedure

1. When the IBM MQ installation completes, the Prepare IBM MQ Wizard window is displayed with a welcome message. To continue, click **Next**
2. If you have run the Prepare IBM MQ wizard before, this step is skipped. If you have not run the Prepare IBM MQ wizard before, the Prepare IBM MQ Wizard window displays a progress bar with the following message:

   ```
   Status: Setting up IBM MQ Configuration
   ```

   Wait until the progress bar completes.
3. The Prepare IBM MQ Wizard window displays a progress bar with the following message:

   ```
   Status: Setting up the IBM MQ Service.
   ```

   Wait until the progress bar completes.
4. IBM MQ attempts to detect whether you must configure IBM MQ for use with Windows Active Directory Server or later domain users. Depending on the results of the detection, IBM MQ does one of the following things:
   - If IBM MQ detects that you need to configure IBM MQ for Windows Active Directory Server or later domain users, the Prepare IBM MQ Wizard window displays a message that starts:

     ```
     IBM MQ does not have the authority to query information about
     your user account
     ```

     Optionally, to see online help about configuring the domain account, select More Information. When you are finished, close the IBM MQ Help Center window to return to the current window.

     Click **Next**, and go to step 5.
   - If you are not installing on a Windows Active Directory Server or later domain server and IBM MQ cannot detect whether you need to configure IBM MQ for Windows Active Directory Server or later domain users, the Prepare IBM MQ Wizard window displays the following message:

     ```
     Are any of the domain controllers in your network running
     Windows 2000 or later domain server?
     ```

     If you select Yes, click **Next**, then go to step 5.

     If you select No, click **Next**, then go to step 9.

     If you select **Don't know**, you cannot continue. Select one of the other options, or click **Cancel** and contact your domain administrator.
   - If IBM MQ detects that you do not need to configure IBM MQ for Windows Active Directory Server or later domain users, go to step 9.

   Note: At any time, you can click **More Information** to view online help about configuring the domain account, or see “Configuring IBM MQ accounts” on page 475. When you are finished, close the IBM MQ Help Center window to return to the current window.
5. The Prepare IBM MQ Wizard window displays the following message:

   ```
   Do you need to configure IBM MQ for users defined on Windows 2000
   or later domain controllers?
   ```

   If you select Yes, click **Next**, then go to step 6.

   If you select No, click **Next**, then go to step 9.

   If you select Don't know, you cannot continue. Select one of the other options, or click **Cancel** and contact your domain administrator.

   Note: At any time, you can click **More Information** to view online help about configuring the domain account, or see “Configuring IBM MQ accounts” on page 475. When you are finished, close the IBM MQ Help Center window to return to the current window.
6. Give the domain user that you obtained from your domain administrator the access 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**.
   c. Click **Add User or Group...** and 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.
7. In the next window, enter the Domain and user ID of the domain user account that you obtained from your domain administrator. Either enter the Password for this account, or select the option **This account does not have a password**. Click **Next**.
8. The Prepare IBM MQ Wizard window displays a progress bar with the following message:

   ```
   Status: Configuring IBM MQ with the special domain user account
   ```

   Wait until the progress bar completes.

   If there are any problems with the domain user account, a further window is displayed. Follow the advice on this window before you continue with this procedure.
9. The Prepare IBM MQ Wizard window displays a progress bar with the following message:

   ```
   Status: Starting IBM MQ services
   ```

   Wait until the progress bar completes.
10. Next, select the options that you require. The Prepare IBM MQ Wizard window displays the following message:

    ```
    You have completed the Prepare IBM MQ Wizard
    ```

    Select the options that you require, then click **Finish**. Select one or more from:
    - **Remove the shortcut to this wizard from the desktop**

      This option is available only if you have previously attempted installation, but you canceled the procedure from the Prepare IBM MQ wizard and you created a desktop shortcut to this wizard. Select this option to remove the shortcut. You do not need it now that you have completed the Prepare IBM MQ wizard.
    - **Launch IBM MQ Explorer**

      The IBM MQ Explorer allows you to view and administer your IBM MQ network.
    - **Launch Notepad to view the release notes**

      The release notes contain information about installing IBM MQ and also late-breaking news that is available after the published documentation is produced.
11. Follow the procedure described in "Checking for problems after installing" on page 474.

Related information:
User rights required for an IBM MQ Windows Service

Checking for problems after installing: Windows

There are some optional tasks that you can use to check the installation if you believe there was a problem, or to verify installation messages after an unattended (silent) installation for example.

About this task

Use these steps as a guide to check the following files for messages:

Procedure

1. `MSI` *`nnnnn`*`.LOG`. This file is in your user `Temp` folder. It is an application log that contains English messages written during installation. The log includes a message indicating whether the installation was successful and complete.

   This file is created if you have set up default logging.
2. If you used the launchpad to install IBM MQ, check `MQv7_Install_YYYY-MM-DDTHH-MM-SS.log` in your user `Temp` folder, where:

   YYYY This is the year that you installed IBM WebSphere MQ Version 7.0

   MM This is the month that you installed IBM MQ, for example this would be 09 if you installed in September

   DD This is the day that you installed IBM MQ

   HH-MM-SS
   This is the time at which IBM MQ was installed

   You can get to your user Temp directory by entering the following command at the command prompt:

   ```
   cd %TEMP%
   ```
3. `amqmjpse.txt`. This file is in the IBM MQ data files folder (default `C:\ProgramData\IBM\MQ`). It is an application log that contains English messages written during installation by the Prepare IBM MQ wizard.

What to do next

1. Verify your installation, as described in *Verifying your IBM MQ installation* for the platform, or platforms, that your enterprise use.

Configuring IBM MQ accounts: Windows

The IBM MQ service and queue managers check that any users attempting to access queue managers or queue manager resources such as queues, have the permission to access them.

Most networked Windows systems are members of a Windows domain where user accounts, other security principals, and security groups are maintained and managed by a directory service, Active Directory, running on a number of domain controllers. IBM MQ checks that only authorized users can access queue managers or queues.

In such networks, IBM MQ queue manager processes access the Active Directory information to find the security group membership of any users attempting to use IBM MQ resources. The accounts under which IBM MQ services run must be authorized to look up such information from the directory. In most Windows domains, local accounts defined at individual Windows servers cannot access directory information, so the IBM MQ services must run under a domain account that has the appropriate permission.

If the Windows server is not a member of a Windows domain or the domain has a reduced security or functional level, then the IBM MQ services can run under a local account that was created during installation.

Assuming that a domain account is needed, provide the information described in the Information for domain administrator to your domain administrator, and ask for one of the special accounts it describes. When you install the product, towards the end of the installation procedure, in the Prepare IBM MQ wizard, you are asked to enter details of this account (domain, user name, and password).

If a domain account is needed and you install IBM MQ without a special account (or without entering its details), many or all parts of IBM MQ do not work, depending upon the particular user accounts involved. Also, IBM MQ connections to queue managers that run under domain accounts on other systems might fail. The account can be changed by running the Prepare IBM MQ wizard and specifying the details of the account to be used.

For information about the user rights required to take advantage of the Active Directory support, see Using Active directory (Windows only).

For information about the user rights required to take advantage of the Kerberos authentication support, see Securing.

Information for domain administrators: Windows

Use this topic to understand how IBM MQ services check the authorization of user accounts attempting to access IBM MQ.

The user account must either have an individual IBM MQ authorization set or belong to a local group that has been authorized. A domain account can also be authorized through membership of a domain group included under an authorized local group through a single level of nesting.

The account under which the IBM MQ services are run must have the ability to query group memberships of domain accounts and have the authority to administer IBM MQ. Without the ability to query group memberships the access checks made by the services fail.

On most Windows domains, with domain controllers running Windows Active Directory, local accounts do not have the required authorization and a special domain user account with the required permissions must be used. The IBM MQ installer must be given the userid and password details so that they can be used to configure the IBM MQ service after the product is installed.

Typically, this special account has the IBM MQ administrator rights through membership of the domain group `DOMAIN\Domain mqm`. The domain group is automatically nested by the installation program under the local mqm group of the system on which IBM MQ is being installed.

See "Creating and setting up domain accounts for IBM MQ" for instructions on creating a suitable domain account.

Note: If an installer configures IBM MQ without a special account, many or all parts of IBM MQ do not work, depending upon the particular user accounts involved, as follows:
- An installer currently logged on with a domain user account is not be able to complete the Default Configuration, and the Postcard application does not work.
- IBM MQ connections to queue managers running under domain accounts on other systems might fail.
- Typical errors include "AMQ8066: Local mqm group not found" and "AMQ8079: Access was denied when attempting to retrieve group membership information for user 'abc@xyz'".

Creating and setting up domain accounts for IBM MQ: Windows

The following information is intended at Domain Administrators. Use this information to create and set up domain accounts for IBM MQ.

About this task

Repeat Steps 1 and 2 on page 477 for each domain that has user names that will install IBM MQ, to create an account for IBM MQ on each domain:

Procedure
1. Create a domain group with a special name that is known to IBM MQ and give members of this group the authority to query the group membership of any account:
   a. Log on to the domain controller as an account with domain administrator authority.
   b. From the Start menu, open Active Directory Users and Computers.
   c. Find the domain name in the navigation pane, right-click it and select **New Group**.
   d. 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.
   e. Leave **Global** clicked as the **Group scope**, or change it to **Universal**. Leave **Security** clicked as the **Group type**. Click **OK**.

f. Follow these steps to assign permissions to the group based on the Windows version of the domain controller:

   On Windows Server 2012 and Windows Server 2012 R2

   1) In the Server Manager, click **Tools** then select **Active Directory Users and Computers** from the list box.
   2) Select **View** > **Advanced Features**
   3) Expand your domain name, then click **Users**
   4) In the Users window, right-click **Domain mqm** > **Properties**
   5) Click **Security** > **Advanced** > **Add...**
   6) Click **Select principle**, then type `Domain mqm` and click **Check names** > **OK**

      The **Name** field is prefilled with the string `Domain mqm` (*domain name*`\Domain mqm`)
   7) In the **Applies to** list, select **Descendant User Objects**
   8) In the **Permissions** list, select the **Read group membership** and **Read groupMembershipSAM** check boxes.
   9) Click **OK** > **Apply** > **OK** > **OK**.

   On Windows Server 2008 and Windows 2008 R2:

   1) In the Server Manager navigation tree, click **Users**.
   2) In the Server Manager action bar, click **View** > **Advanced features**
   3) In the Users window, right-click **Domain mqm** > **Properties**
   4) Click **Security** > **Advanced** > **Add**, then type `Domain mqm` and click **Check names** > **OK**

      The **Name** field is prefilled with the string `Domain mqm` (*domain name*`\Domain mqm`)
   5) Click **Properties**. In the **Apply to** list, select **Descendant User Objects**
   6) In the **Permissions** list, select the **Read group membership** and **Read groupMembershipSAM** check boxes.
   7) Click **OK** > **Apply** > **OK** > **OK**.

2. Create one or more accounts, and add them to the group:
   a. In **Active Directory Users and Computers**, create a user account with a name of your choosing and add it to group `Domain mqm` (or a group that is a member of the local `mqm` group).
   b. Repeat for all the accounts you want to create.
3. Repeat Steps 1 on page 476 and 2 for each domain that has user names that will install IBM MQ, to create an account for IBM MQ on each domain.
4. Use the accounts to configure each installation of IBM MQ:
   a. Either use the same domain user account (as created in Step 1 on page 476) for each installation of IBM MQ, or create a separate account for each one, adding each to the `Domain mqm` group (or a group that is a member of the local `mqm` group).
   b. When you have created the account or accounts, give one to each person configuring an installation of IBM MQ. They must enter the account details (domain name, user name, and password) into the Prepare IBM MQ wizard. Give them the account that exists on the same domain as their installing userid.
   c. When you install IBM MQ on any system on the domain, the IBM MQ installation program detects the existence of the `Domain mqm` group on the LAN, and automatically adds it to the local `mqm` group. (The local `mqm` group is created during installation; all user accounts in it have authority to manage IBM MQ). Thus all members of the "`Domain mqm`" group will have authority to manage IBM MQ on this system.
   d. However, you do still need to provide a domain user account (as created in Step 1 on page 476) for each installation, and configure IBM MQ to use it when making its queries. The account details must be entered into the Prepare IBM MQ wizard that runs automatically at the end of installation (the wizard can also be run at any time from the **start** menu).

5. Set the password expiry periods:
   - If you use just one account for all users of IBM MQ, consider making the password of the account never expire, otherwise all instances of IBM MQ will stop working at the same time when the password expires.
   - If you give each user of IBM MQ their own user account you will have more user accounts to create and manage, but only one instance of IBM MQ will stop working at a time when the password expires.

   If you set the password to expire, warn the users that they will see a message from IBM MQ each time it expires - the message warns that the password has expired, and describes how to reset it.
6. 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.

Using the Default Configuration wizard: ▶ Windows

You can use the Default Configuration wizard to add the first configured queue manager to this system. This enables you to connect easily with other queue managers in the same IBM MQ cluster.

About this task

You can use the Default Configuration wizard to create, view, or alter your default configuration. You can also use this wizard to alter or display details of an existing queue manager that was created by the default configuration.

For a new installation of IBM MQ, creating a default configuration enables you to explore features of IBM MQ using the Postcard application, and the IBM MQ Explorer.

The Postcard application provides a fast and simple way to verify that your IBM MQ installation completed successfully. It uses the default queue manager that is created during the default configuration. If you want to use the Postcard application for verification, and you do not have any existing queue managers, run the Default Configuration wizard first.

If you have migrated existing queue managers, or created any queue managers since installing IBM MQ, you might not want to run the Default Configuration wizard. This is because you cannot create the default configuration if other queue managers already exist. If you have previously created any other queue managers on this system and you still want to set up a default configuration, you must delete them before you run the Default Configuration wizard.

Start the Default Configuration wizard by selecting **Create the Default Configuration** on the Welcome to IBM MQ Explorer Content view page.

Using the Welcome to IBM MQ Explorer Content view page: ▶ Windows

The Welcome to IBM MQ Explorer Content view page points you to any relevant applications, documentation, tutorials, and education. This page is displayed the first time you launch IBM MQ Explorer.

You can use the items in the Welcome to IBM MQ Explorer Content view page to explore the facilities in IBM MQ. This page is launched the first time the IBM MQ Explorer is launched. The Welcome page can be viewed at any time from the IBM MQ Explorer by clicking **IBM MQ** in the Navigator view. There are links to the following subjects from this page:

Create the Default Configuration

Allows you to add a configured queue manager to this system for connecting easily with other queue managers in the same IBM MQ cluster. You can also use it to alter or display details of an existing queue manager created by the default configuration. This feature is available only using TCP/IP.

Note: If you migrated existing queue managers, or if you have created any queue managers after you installed IBM MQ, you might not want to use this facility. This is because you can only set up a default configuration if there are no queue managers already, and you might not want to delete your existing queue managers.

Launch Postcard

Allows you to try out IBM MQ messaging quickly and easily. You can send a message either to your own machine or to another named user's machine. It is described in detail in "Verifying a server-to-server installation using the Postcard application on Windows" on page 503.

Using the Help Center: ▶ Windows

The Help Center gives you access to all task-oriented help, information on the IBM website, and a link to the IBM MQ product documentation.

The IBM MQ Help Center can be accessed from the IBM MQ Explorer by selecting **Help** > **Help Contents**.

Converting a trial license on Windows

▶ Windows

Convert a trial license to a full license without reinstalling IBM MQ.

When the trial license expires, the "count-down" displayed by the `strmqm` command informs you the license has expired, and the command does not run.

Before you begin

1. IBM MQ is installed with a trial license.
2. You have access to the installation media of a fully licensed copy of IBM MQ.

About this task

Run the **setmqprd** command to convert a trial license to a full license.

If you do not want to apply a full license to your trial copy of IBM MQ, you can uninstall it at any time.

Procedure

1. Obtain the full license from the fully licensed installation media.

   The full license file is `amqpcert.lic`. On Windows it is in the `\`*MediaRoot*`\licenses` directory on the installation media. It is installed into the `bin` directory on the IBM MQ installation path.
2. Run the **setmqprd** command from the installation that you are upgrading:

   ```
   MQ_INSTALLATION_PATH\bin\setmqprd \MediaRoot\licenses\amqpcert.lic
   ```

Related information:

setmqprd

Displaying messages in your national language on Windows systems

Windows

To display messages from a different national language message catalog, you must either set the **MQS_FORCE_NTLANGID** environment variable, or change a regional setting.

About this task

Messages in U.S. English are automatically installed with IBM MQ

Messages in the national languages that IBM MQ supports are automatically installed. Messages are displayed in the national language, based on the following order:

1. The value of the **MQS_FORCE_NTLANGID** environment variable, if set.
2. The regional format of the user that is displaying the message, if the language specified by the regional format is supported by IBM MQ.
3. The Administrative system locale if the language specified by the system locale is supported by IBM MQ.
4. US English, if no other supported language can be determined.

Note: The queue manager is usually launched by a service on the machine, and hence is running under its own user account (for example MUSR_MQADMIN) or a specific domain account provided during install time. See Security on Windows for more information.

If you require messages in a language other than the one associated with the regional format of a user account, perform the following steps:

Procedure

1. Globally set the **MQS_FORCE_NTLANGID** environment variable, to the language identifier of the desired language, for messages displayed by the queue manager. You should set the **MQS_FORCE_NTLANGID** system wide. Otherwise, every user displaying messages needs to have the environment variable set individually.

   The language identifier values, represented in hexadecimal notation, are listed in the following Microsoft document: Language Identifier Constants and Strings
2. Reboot machines where queue managers are running as a service, for the environment variable to take effect.

Installing an IBM MQ client on Windows

Windows

This topic describes how to install IBM MQ client on Windows systems. This procedure can be used for installing a first or a subsequent installation.

Before you begin

To install an IBM MQ client, you must be logged on to Windows as an administrator.

About this task

Follow these instructions to perform an interactive compact, typical, or custom installation of IBM MQ. To view all the installation types and the features that are installed with each option consult Features installed with each type of interactive installation.

Attention: From IBM MQ Version 9.0, if you are using **`msiexec`** to install the client, the installation is automatically set to be the primary installation.

Procedure

1. Access the IBM MQ installation image. The location might be the mount point of the DVD, a network location, or a local file system directory.
2. Locate `setup.exe` in the Windows directory of the IBM MQ installation image.
   - From a DVD, this location might be:
     `E:\Windows\setup.exe`
   - From a network location, this location might be:
     `m:\instmqs\Windows\setup.exe`
   - From a local file system directory, this location might be:
     `C:\instmqs\Windows\setup.exe`
3. Double-click the **Setup** icon to start the installation process. It is possible to run either by:
   - Running `setup.exe` from the command prompt. Or
   - Double-clicking `setup.exe` from Windows Explorer.

   If you are installing on a Windows system with UAC enabled, accept the Windows prompt to allow the launchpad to run as elevated. During installation, you might also see Open File - Security Warning dialog boxes that list International Business Machines Limited as the publisher. Click **Run** to allow the installation to continue.

   The IBM MQ Installation window is displayed.
4. Continue to follow the instructions as shown on screen.

Results

A new sample IBM MQ MQI client configuration file is created in the IBM MQ installation directory (for example `C:\Program Files\IBM\MQ\`, by the IBM MQ MQI client package, during installation, but only if this file does not exist. This file contains the `ClientExitPath` stanza. An example `mqclient.ini` file is shown in Configuring a client using a configuration file.

Note:

If you are using a common configuration file for multiple clients, either in the IBM MQ installation directory or in another location using the MQCLNTCF environment variable, you must grant read access to all user identifiers under which the IBM MQ client applications run. If the file cannot be read, the failure is traced and the search logic continues as if the file had not existed.

What to do next

- If you have chosen this installation to be the primary installation on the system, when using `setup.exe`, you must now set it as the primary installation. Enter the following command at the command prompt:

```
MQ_INSTALLATION_PATH\bin\setmqinst -i -p MQ_INSTALLATION_PATH
```

  You can have only one primary installation on a system. If there is already a primary installation on the system, you must unset it before you can set another installation as the primary installation. For more information, see Changing the primary installation.
- You might want to set up the environment to work with this installation. You can use the **setmqenv** or **crtmqenv** command to set various environment variables for a particular installation of IBM MQ. For more information, see setmqenv and crtmqenv.
- For instructions on how to verify your installation, see "Testing communication between a client and a server on Windows" on page 512.

Related concepts:

"Modifying a client installation using Add/Remove Programs" on page 492
On some versions of Windows, you can modify an installation by using Add/Remove Programs.

Related tasks:

"Installing a client using msiexec"
IBM MQ on Windows uses the MSI technology to install software. MSI provides both an interactive installation and a non interactive installation.

"Installing a client using the MQParms command" on page 489
You can use the **MQParms** command to invoke installation or uninstallation of an IBM MQ client.

"Uninstalling IBM MQ on Windows" on page 514
You can uninstall the IBM MQ MQI clients and servers on Windows systems by using the control panel, the command line (**msiexec**), **MQParms**, or by using the installation media, in which case you can optionally remove queue managers as well.

Installing a client using msiexec

Windows

IBM MQ on Windows uses the MSI technology to install software. MSI provides both an interactive installation and a non interactive installation.

About this task

IBM MQ on Windows uses the MSI technology to install software. MSI provides both an interactive installation and a non interactive installation. An interactive installation displays panels and ask questions.

The **msiexec** command uses parameters to give MSI some or all of the information that can also be specified through panels during an interactive installation. This means that a user can create a reusable automated or semi-automated installation configuration. Parameters can be given through the command line, a transform file, a response file, or a combination of the three.

Procedure

To install using msiexec, at the command line, enter the **msiexec** command in the following format:

```
msiexec parameters [USEINI="response-file"] [TRANSFORMS="transform_file"]
```

Where:

parameters
: are either command-line parameters preceded by a / character, or property=value pairs (if using both forms of parameter always put the command-line parameters first). For further information, see "Specifying command line parameters with msiexec."

For an unattended installation, you must include the /q or /qn parameter in the command line. Without this parameter, the installation is interactive.

Note: You must include the **/i** parameter and the file location of the IBM MQ installer package.

response-file
: is the full path and file name of the file that contains the [Response] stanza and the required property=value pairs, for example `C:\MyResponseFile.ini`. An example response file, `Response.ini`, is supplied with IBM MQ. This file contains default installation parameters. For further information, see "Using a response file with msiexec" on page 484.

transform_file
: is the full path and file name of a transform file. For further information, see "Using transforms with msiexec for client installation" on page 487 and "Choosing MSI Instance IDs for multiple server installations" on page 461.

Note: For a silent installation to succeed, the AGREETOLICENSE="yes" property must be defined either on the command line or in the response file.

Results

After the command has been entered, the command prompt immediately reappears. IBM MQ is installing as a background process. If you have entered parameters to produce a log, check this file to see how the installation is progressing. If the installation completes successfully, you see the message `Installation operation completed successfully` in the log file.

Specifying command line parameters with msiexec:
About this task

The **msiexec** command can accept two types of parameters on the command line, as follows:

- Standard command line parameters, preceded by a / character.

  For a table of the **msiexec** command line parameters, see the MSDN Command-Line Options web page.
- Property=value pair parameters on the command line. All the parameters available for use in a response file can be used on the command line, for a list of these, see Table 70 on page 485. In addition there are some extra property=value pair parameters that are only for use on the command line, for details see Table 69 on page 484.

  When using the property=value pair parameters note that:
  - Property strings must be in uppercase.
  - Value strings are not case-sensitive, except for feature names. You can enclose value strings in double quotation marks. If a value string includes a blank, enclose the blank value string in double quotation marks.
  - For a property that can take more than one value, use the format:

    ```
    ADDLOCAL="Server,Client"
    ```
  - For properties taking paths and filenames, for example PGMFOLDER, you must supply the paths as absolute paths and not relative; that is, `C:\folder\file` and not `.\folder\file`.

  When using property=value pair and command line parameters with the **msiexec** command, enter command line parameters first.

  If a parameter is specified both on the command line and in a response file, the setting on the command line takes precedence.

Example

A typical example of an **msiexec** command is:

```
msiexec /i "path\MSI\IBM MQ.msi" /l*v c:\install.log
/q TRANSFORMS="1033.mst" AGREETOLICENSE="yes" ADDLOCAL="Client"
```

A typical example of an **msiexec** command when you are installing a second copy of the IBM MQ product is:

```
msiexec /i "path\MSI\IBM MQ.msi" /l*v c:\install.log
/q TRANSFORMS=":InstanceId2.mst;1033.mst" AGREETOLICENSE="yes"
ADDLOCAL="Client" MSINEWINSTANCE=1
```

The following table shows the parameters which can only be provided on the command line and not in a response file.

Table 69. msiexec property=value parameters

Property	**Values**	**Meaning**
USEINI	*path* \ *file_name*	Use the specified response file. See "Using a response file with msiexec"
SAVEINI	*path* \ *file_name*	Generate a response file during installation. The file contains those parameters selected for this installation that a user might make during an interactive installation.
ONLYINI	1|yes| ""	1, yes or any value other than null. End the installation before updating the target system, but after generating a response file, if this is specified. "". Continue the installation and update the target system (the default).
TRANSFORMS	:InstanceId *x*.mst| *path* \ *file_name* | :InstanceId *x*.mst; *path* \ *file_name*	The :InstanceId *x*.mst value is only required for a subsequent installation of IBM WebSphere MQ Version 7.1 or later. The *path* \ *file_name* specifies what transform (.mst) files must be applied to the product. For example, "1033.mst" specifies the supplied U.S. English transform file.
MSINEWINSTANCE	1	This property is only required for subsequent installations of IBM WebSphere MQ Version 7.1 or later.
REMOVEFEATURES	yes	Required with value "yes" for a silent installation, otherwise ignored. Allows obsolete features, no longer part of IBM MQ, to be deleted.

Using a response file with msiexec:
About this task

You can use the **msiexec** command with a parameter which specifies additional properties are defined in a response file. You can combine the `msiexec` command-line parameters described in "Specifying command line parameters with msiexec" on page 483.

A response file is an ASCII text file, with a format like a Windows `.ini` file, that contains the stanza [Response]. The [Response] stanza contains some or all the parameters that would normally be specified as part of an interactive installation. The parameters are given in a property=value pair format. Any other stanzas in the response file are ignored by **`msiexec`**. An example response file, `Response.ini`, is supplied with IBM MQ. It contains the default installation parameters.

Procedure

A typical example of an `msiexec` command is: `msiexec /i "`*`path`*`\MSI\IBM MQ.msi" /l*v c:\install.log TRANSFORMS="1033.mst" USEINI="C:\MQ\Responsefile"`
If a parameter is specified both on the command line and in a response file, the setting on the command line takes precedence. All the parameters available for use in a response file can also be used on the command line, for a list of these see Table 70.
In the response file, all text is in English, and comments begin with a **;** character.
For information about creating a response file, see "Creating a response file for server installation" on page 463.

Example

An example of a typical response file:

```
[Response]
PGMFOLDER="c:\mqm"
DATFOLDER="c:\mqm\data"
AGREETOLICENSE="yes"
ADDLOCAL="Client"
REMOVE="Toolkit"
```

Table 70. Response file parameters

Property	Values	Meaning
PGMFOLDER	*path*	Folder for the IBM MQ program files. For example, `c:\mqm`.
DATFOLDER	*path*	Folder for the IBM MQ data files. For example, `c:\mqm\data`.
USERCHOICE	0|no	If the command line or response file specifies parameters to install features, a dialog can be displayed to prompt you to accept the preselected options, or review and possibly change them. 0 or no. Suppresses display of the dialog. Anything else. Dialog is displayed and you can amend the options. Not used for a silent installation.
AGREETOLICENSE	yes	Accept the terms of the license. Set to `yes` before a silent installation. If the installation is not silent, this parameter is ignored.

Table 70. Response file parameters (continued)

Property	Values	Meaning
ADDLOCAL	*feature, feature, All*| ""	A comma-separated list of features to install locally. For a list of valid feature names, see "IBM MQ features for Windows systems" on page 436. All installs all features "" installs the typical features. If you do not want a feature use `REMOVE="`*`feature`*`"` **Note:** If this is a new installation the typical features (Client, Java, .NET Messaging, and Development Toolkit) are installed by default irrespective of the feature list provided in the ADDLOCAL property. If you do not want a feature use `REMOVE="`*`feature`*`"`
REMOVE	*feature, feature,* |All| ""	A comma-separated list of features to remove. For a list of valid feature names, see "IBM MQ features for Windows systems" on page 436. All uninstalls all features "" uninstalls no features (the default).
INSTALLATIONDESC	"Description of installation"	Sets the installation description from the command line. Subject to the documented installation description length limitations
INSTALLATIONNAME	[INSTALLATION0,]Name	Sets the installation name from the command line. Subject to the documented installation name character and length limitations. **Note:** Supply INSTALLATION0,Name only when upgrading from a release earlier than IBM WebSphere MQ Version 7.1.
MAKEPRIMARY	0|1| ""	Makes the installation primary, if possible, or removes the primary flag. 1 = Make primary, 0 = Make non-primary, - use default algorithm **Note:** This option is ignored if a release earlier than IBM WebSphere MQ Version 7.1 is installed, or if another Version 7.1 or later installation is present and set as the primary.

Related tasks:

"Installing a client using the MQParms command" on page 489
You can use the **`MQParms`** command to invoke installation or uninstallation of an IBM MQ client.

Related reference:

"Using transforms with msiexec for client installation" on page 487

Choosing MSI Instance IDs for multiple client installations: Windows

For multiple silent installations, for each version that is installed you must find an MSI instance ID that is available to use for that installation.

About this task

In order to support silent, or non-interactive, multiple installations, you need to find out whether the instance ID you want to use is already in use or not and choose the appropriate one. For each installation media (for example, each client and server), Instance ID 1 is the default ID which is used for single installations. If you want to install alongside Instance ID 1 you need to specify which instance you want to use. If you have already installed instance 1, 2, and 3 then you need to find out what the next available instance is, for instance, Instance ID 4. Similarly, if instance 2 has been removed, you need to find out that there is a gap that can be reused. You can find out which Instance ID is currently in use by using the **dspmqinst** command.

Procedure

1. Type **dspmqinst** to find a free MSI Instance in the media being installed by reviewing the MSIMedia and MSIInstanceId values for the versions already installed. For example:

   ```
   InstName: Installation1
   InstDesc:
   Identifier:    1
   InstPath:      C:\Program Files\IBM\MQ
   Version:       9.0.0.0
   Primary:       Yes
   State:         Available
   MSIProdCode:   {74F6B169-7CE6-4EFB-8A03-2AA7B2DBB57C}
   MSIMedia:      9.0 Server
   MSIInstanceId: 1
   ```

2. If MSI Instance ID 1 is in use and you want to use MSI Instance ID 2, the following parameters must be added to the msiexec call:

   ```
   MSINEWINSTANCE=1 TRANSFORMS=":instanceId7.mst;1033.mst"
   ```

What to do next

For multiple installations, the **INSTALLATIONNAME** or **PGMFOLDER** must be supplied as an additional parameter on any non-interactive installation command. Supplying the **INSTALLATIONNAME** or **PGMFOLDER** ensures that you do not work with the wrong installation in case you omit or incorrectly specify the **TRANSFORMS** parameter.

Using transforms with msiexec for client installation: Windows

MSI can use transforms to modify an installation. During IBM MQ installation, transforms can be used to support different national languages. IBM MQ is supplied with transform files in the `\MSI` folder of the client image. These files are also embedded in the IBM MQ Windows installer package, `IBM MQ.msi`.

On the **msiexec** command line, you can specify the required language by using the TRANSFORMS property in a property=value pair. For example:

```
TRANSFORMS="1033.mst"
```

You can also specify the full path and file name of the transform file. Again, the quotation marks surrounding the value are optional. For example:

```
TRANSFORMS="D:\Msi\1033.mst"
```

Table 71 on page 488 shows the locale identifier, language, and the transform file name to use in the **msiexec** command line.

You might need to merge transforms to install multiple installations of the same version, for example:

```
TRANSFORMS=":InstanceId2.mst;D:\Msi\1033.mst"
```

You can also specify the required language by using the MQLANGUAGE property with the **MQParms** command. For information about the msiexec property=value parameters, see “MQParms parameter file - client installation” on page 490.

Parameters

Table 71. Supplied transform files for various language support. This table shows the supplied transform files, the resulting language, and the numeric value to use in the **msiexec** command line.

Language	Transform File name	Value
U.S. English	1033.mst	1033
German	1031.mst	1031
French	1036.mst	1036
Spanish	1034.mst	1034
Italian	1040.mst	1040
Brazilian Portuguese	1046.mst	1046
Japanese	1041.mst	1041
Korean	1042.mst	1042
Simplified Chinese	2052.mst	2052
Traditional Chinese	1028.mst	1028
Czech	1029.mst	1029
Russian	1049.mst	1049
Hungarian	1038.mst	1038
Polish	1045.mst	1045

Creating a response file for client installation: Windows

A response file is used with **msiexec** on a client. You can create it in three ways.

About this task

A response file is used with the **msiexec** command. For further information, see "Using a response file with msiexec" on page 457.

Procedure

There are three ways to create a response file for installation:
- Copy and edit the file `Response.ini` that is supplied on the IBM MQ Windows Server DVD, using an ASCII file editor.
- Create your own response file using an ASCII file editor.
- Use the **msiexec** command with the **SAVEINI** (and optionally, the **ONLYINI**) command line parameters to generate a response file that contains the same installation options. See Table 63 on page 457.

Example

A typical example of using **msiexec** with the **SAVEINI** parameter is here:

```
msiexec /i "path\IBM MQ.msi" /q SAVEINI="response_file"
TRANSFORMS="1033.mst" AGREETOLICENSE="yes"
```

Installing a client using the MQParms command: Windows

You can use the **MQParms** command to invoke installation or uninstallation of an IBM MQ client.

Before you begin

The **MQParms** command can use parameters on a command line, or those specified in a parameter file. The parameter file is an ASCII text file that contains the parameter values that you want to set for the installation. The **MQParms** command takes the specified parameters and generates the corresponding **msiexec** command line.

This means that you can save all the parameters that you want to use with the **msiexec** command in a single file.

If you are running IBM MQ on Windows systems with User Account Control (UAC) enabled, you must invoke the installation with elevated privileges. If you are using the Command prompt or IBM MQ Explorer elevate privileges by using a right-click to start the program and selecting **Run as administrator**. If you try to run the `MQParms` program without using elevated privileges, the installation fails with an error of AMQ4353 in the installation log.

For silent operations, this must include the **/q** or **/qn** parameter, either on the command line, or in the [MSI] stanza of the parameter file. You must also set the AGREETOLICENSE parameter to "yes".

You can specify many more parameters in the parameter file that you use with the `MQParms` command than you can in the response file that you use directly with the **msiexec** command. Also, as well as parameters that the IBM MQ installation uses, you can specify parameters that can be used by the Prepare IBM MQ wizard.

If you do not complete the Prepare IBM MQ Wizard directly after IBM MQ installations or if for any reason your machine is rebooted between completing IBM MQ installation and completing the Prepare IBM MQ Wizard, ensure that the wizard is run with Administrator privilege afterward, otherwise the installation is incomplete, and might fail. You might also see Open File - Security Warning dialog boxes that list International Business Machines Limited as the publisher. Click **Run** to allow the wizard to continue

An example of the file `MQParms.ini` is supplied with IBM MQ. This file contains default installation parameters.

There are two ways to create a parameter file for installation:
- Copy and edit the file `MQParms.ini` that is supplied with the product, using an ASCII file editor.
- Create your own parameter file using an ASCII file editor.

About this task

To invoke installation using the `MQParms` command:

Procedure

1. From a command line, change to the root folder of the IBM MQ client CD (that is, the location of the file MQParms.exe).
2. Enter the following command:

   `MQParms [ `*`parameter_file`*` ] [ `*`parameters`*` ]`

   where:

   parameter_file

   is the file that contains the required parameter values. If this file is not in the same folder as

MQParms.exe, specify the full path and file name. If you do not specify a parameter file, the default is MQParms.ini. For further details, see "MQParms parameter file - client installation."

parameters
: are one or more command-line parameters, for a list of these, see the MSDN Command-Line Options web page.

Example

A typical example of an MQParms command is:

```
MQParms "c:\MyParamsFile.ini" /l*v c:\install.log
```

If you specify a parameter both on the command line and in the parameter file, the setting on the command line takes precedence.

If you do not specify `/i`, `/x`, `/a`, or `/j`, MQParms defaults to standard installation using the IBM MQ Windows Installer package, IBM IBM MQ.msi. That is, it generates the following part of the command line:

```
/i " current_folder \MSI\IBM MQ.msi"
```

MQParms parameter file - client installation: Windows

A parameter file is an ASCII text file that contains sections (stanzas) with parameters that can be used by the **MQParms** command. Typically, this is an initialization file such as `MQParms.ini`.

The **MQParms** command takes parameters from the following stanzas in the file:

[MSI] Contains general properties related to how the **MQParms** command runs and to the installation of IBM MQ.

The properties that you can set in this stanza are listed in "Installing a client using msiexec" on page 482, and Table 72 on page 491.

`MQParms` ignores any other stanzas in the file.

The stanza parameters are in the form property=value, where property is always interpreted as uppercase, but value is case sensitive. If a value string includes a blank, it must be enclosed in double quotation marks. Most other values can be enclosed in double quotation marks. Some properties can take more than one value, for example:

```
ADDLOCAL="Server,Client"
```

To clear a property, set its value to an empty string, for example:

```
REINSTALL=""
```

The following tables show the properties that you can set. The default is shown in bold.

For the [MSI] stanza, you can enter standard MSI command line options and properties. For example:

```
- /q
- ADDLOCAL="client"
- REBOOT=Suppress
```

Refer to Table 72 on page 491, and Table 73 on page 491 for the properties used to install IBM MQ.

Table 72 on page 491 shows additional properties in the stanza that affect how the `MQParms` command runs, but that do not affect the installation.

Table 72. Properties used by MQParms in the MSI stanza

Property	Values	Description
MQPLOG	*path* | *file_name*	**MQParms** generates a text log file with the specified name and location.
MQPLANGUAGE	**system** |user| *transform_value* |existing	The installation language. system. Install using the language of the default system locale (the default). user. Install using the language of the default locale of the user. *transform_value*. Install using the language specified by this value. See Table 73. existing. If MQ already exists on the system, the same language will be used by default, otherwise system is used.
MQPSMS	**0** |no	0 or no. **MQParms** does not wait for the `msiexec` command to end (the default). Any other value. **MQParms** waits for the `msiexec` command to end.
MQPINUSE	**0** |1	If MQPINUSE is set to 1, **MQParms** continues installing even if IBM MQ files are in use. If this option is used a reboot will be required to complete the installation.

Table 73. Valid values for the MQPLANGUAGE property

Language	Valid values		
U.S. English	English	en_us	1033
German	German	de_de	1031
French	French	fr_fr	1036
Spanish	Spanish	es_es	1034
Italian	Italian	it_it	1040
Brazilian Portuguese		pt_br	1046
Japanese	Japanese	ja_jp	1041
Korean	Korean	ko_kr	1042
Simplified Chinese		zh_cn	2052
Traditional Chinese		zh_tw	1028
Czech	Czech	cs_cz	1029
Russian	Russian	ru_ru	1049
Hungarian	Hungarian	hu_hu	1038
Polish	Polish	pl_pl	1045

A typical example of a parameter file is:

```
[MSI]
MQPLANGUAGE=1033
MQPLOG=%temp%\MQParms.log
MQPSMS=no
```

```
ADDLOCAL=CLIENT
/m miffile
REMOVE=""
/l*v c:\install.log
```

Modifying a client installation on Windows

Windows

You modify the installation when an IBM MQ for Windows client is installed and you want to remove or install some IBM MQ client features.

1. Insert the IBM MQ client DVD into the DVD drive.
2. If autorun is installed, the installation process starts.

   Otherwise, double-click **Setup** in the root folder of the DVD to start the installation process.

   The IBM MQ client Setup window is displayed. Click **Next** to continue.
3. Select **Modify**, then click **Next**.

   The Features panel is displayed.
4. To change the installation of a feature:
   a. Click the symbol next to the feature name to display a menu.
   b. Select the required option from:
      - Install this feature
      - Install this feature and all its subfeatures (if any)
      - Do not install this feature (remove if already installed).

      The symbol next to the feature name changes to show the current installation option.
5. When your selections are complete, click **Next**.
6. The IBM MQ client Setup window displays a summary of the installation you selected.

   To continue, click **Modify**.
7. Wait until the progress bar is complete.

   When the IBM MQ client is successfully installed, the IBM MQ client Setup window displays the following message: `Installation Wizard Completed Successfully`

   Click **Finish** to close the window.

Modifying a client installation using Add/Remove Programs: Windows

On some versions of Windows, you can modify an installation by using Add/Remove Programs.

For Windows 7 follow these steps.

1. From the Windows taskbar, select **Start** > **Control Panel**.
2. Select **Add/Remove Programs**.
3. Select **IBM MQ**.
4. Select **Change**.

   The IBM MQ Setup window with the Program Maintenance panel is displayed. Follow the procedure for modifying the installation by using the process from step 3 to the end.

   For Windows 8, the **Add/Remove Programs** option uninstalls the whole product.

   You need to run the `setup.exe` file from the original installation media to make any modifications to the installation.

Modifying a client installation silently using msiexec: Windows

You can use msiexec to modify an IBM MQ client installation.

To silently modify an IBM MQ client installation using msiexec, follow the instructions on the installation pages, but set the ADDLOCAL parameter to include the features you want to add, and set the REMOVE parameter to the features you want to remove.

For example if you used ADDLOCAL="JavaMsg" and REMOVE="" it would modify the installation to include the Java Messaging and Web Services feature.

The instructions for msiexec begin here: "Installing a client using msiexec" on page 482

Modifying a client installation silently using MQParms: Windows

You can use the **`MQParms`** command to modify an IBM MQ client installation.

To silently modify an IBM MQ client installation using **`MQParms`**, follow the instructions on the installation pages, but set the ADDLOCAL parameter to include the features you want to add, and set the REMOVE parameter to the features you want to remove.

For example if you used ADDLOCAL="JavaMsg" and REMOVE="" it would modify the installation to include the Java Messaging and Web Services feature.

For details of the **`MQParms`** command, see "Installing a client using the MQParms command" on page 489.

Redistributable clients on Windows

Windows

The Windows 64-bit image is shipped in a `Win64.zip` file.

File names

The archive or .zip file names describe the file contents and equivalent maintenance levels. For example, in IBM MQ Version 8.0.0, Fix Pack 4 the client images are available under the following file names:

Windows
`8.0.0.4-WS-MQC-Redist-Win64.zip`

Java - all platforms
`8.0.0.4-WS-MQC-Redist-Java.zip`

Choosing the runtime files to distribute with an application

A script file named **`genmqpkg`** is provided by the redistributable client under the `bin` directory.

You can use the **`genmqpkg`** script to generate a smaller subset of files that are tailored to the needs of the application, for which the files are intended to be distributed.

You are asked a series of interactive `Yes` or `No` questions to determine the runtime requirements for an IBM MQ application.

Finally, **`genmqpkg`** asks you to supply a new target directory, where the script duplicates the required directories and files.

Important: IBM support is only able to provide assistance with the full, unmodified set of files contained within the redistributable client packages.

Other considerations

There is a minor change to the default data path of a non-installed client, and now the path is, on:

Windows
: `%HOMEDRIVE%%HOMEPATH%\IBM\MQ\data`

A redistributable client runtime co-exists with a full IBM MQ client or server installation, provided that they are installed in different locations.

Important: Unpacking a redistributable image into the same location as a full IBM MQ installation is not supported.

Classpath changes

The classpath used by **`dspmqver`**, **`setmqenv`**, and **`crtmqenv`** commands, add the `com.ibm.mq.allclient.jar` to the environment, immediately following the `com.ibm.mq.jar` and `com.ibm.mqjms.jar`.

An example of **`dspmqver`** output from the redistributable client on Windows:

```
Name:        IBM MQ
Version:     8.0.0.4
Level:       p800-804-L150909
BuildType:   IKAP - (Production)
Platform:    IBM MQ for Windows (x64 platform)
Mode:        64-bit
O/S:         Windows 7 Professional x64 Edition, Build 7601: SP1
InstName:    MQNI08000004
InstDesc:    IBM MQ V8.0.0.4 (Redistributable)
Primary:     No
InstPath:    C:\Users\johndoe\Desktop\Redist
DataPath:    C:\Users\johndoe\IBM\MQ\data
MaxCmdLevel: 802
```

Related concepts:
"Redistributable clients" on page 212
The IBM MQ redistributable client is a collection of runtime files that are provided in a `.zip` or `.tar` file that can be redistributed to third parties under redistributable license terms, which provides a simple way of distributing applications and the runtime files that they require in a single package.

.NET application runtime - Windows only

Windows

Considerations when using the .NET application.

The runtime DLL files laid down in the *redistributable* images on Windows for .NET applications are normally registered with the global assembly cache (GAC) by a user with system administrator privileges, when installing the primary installation. However, this severely limits the benefits of redistribution.

The *redistributable* package on the Windows platform does not provide any tooling to register DLLs with the GAC, so .NET applications have to locate the appropriate assemblies by other means. There are two options that work in this situation.

Probing

After checking the GAC, the .NET runtime attempts to locate required assemblies through probing. The first location checked is the application base, which is the root location where the application is being run. See the information on *How the Runtime Locates Assemblies* on the Microsoft Web site for more information.

Note that when using this approach, the maintenance level of the assemblies used when building the .NET application must match those used at runtime - for example an application built at IBM MQ Version 8.0.0, Fix Pack 4 must be run with the IBM MQ Version 8.0.0, Fix Pack 4 redistributable client runtime.

Using this approach, a .NET application placed in the `\bin` directory alongside the IBM MQ assemblies picks up assemblies from a primary IBM MQ installation (if one exists), falling back to the redistributable copies.

1. Compile the .NET application under a full IBM MQ installation, that is `csc \t:exe \r:System.dll \r:amqmdnet.dll \lib: \out:nmqwrld.exe nmqwrld.cs`.
2. Copy the `exe` file in the redistributable client zip file into the `\bin` directory.

DEVPATH environment variable

An alternative, that allows your application to be built, distributed, extracted and run as previously, is to use DEVPATH to locate the required assemblies. Unlike with the probing approach, this option overrides any matching assemblies from the GAC. However it is for this reason that Microsoft discourages its use in a production environment.

This approach can be effective where there is a possibility that a full IBM MQ installation is installed on the client. However, there is a good reason to always use the redistributable assemblies.

1. Compile the .NET application under a full IBM MQ installation, that is `csc \t:exe \r:System.dll \r:amqmdnet.dll \lib: \out:nmqwrld.exe nmqwrld.cs)`
2. Copy the `exe` file into, or alongside, the redistributable client zip file.
3. In the same directory as the `exe`, create an application configuration file with the name of the `exe` file suffixed by `.config`, that is `nmqwrld.exe.config` with the following contents:
   ```
   <configuration>
       <runtime>
         <developmentMode  developerInstallation="true" />
       </runtime>
   </configuration>
   ```
4. Call **`setmqenv -s`** and set the *DEVPATH* environment variable to specify the `\bin` directory from the redistributable image before running the application, that is:
   ```
   set DEVPATH=%MQ_INSTALLATION_PATH%\bin
   ```

Starting and stopping trace for the .NET redistributable managed client

You generate trace for the .NET redistributable managed client in the same way as for the stand-alone .NET client. For more information, see Using the stand-alone IBM MQ .NET client.

More information on .NET

For more information on .NET, see Writing and deploying IBM MQ .NET programs.

Related concepts:

"Redistributable clients" on page 212
The IBM MQ redistributable client is a collection of runtime files that are provided in a `.zip` or `.tar` file that can be redistributed to third parties under redistributable license terms, which provides a simple way of distributing applications and the runtime files that they require in a single package.

Verifying an IBM MQ installation on Windows

Windows

The topics in this section provide instructions on how to verify a server or a client installation of IBM MQ on Windows systems.

About this task

You can verify a local (stand-alone) server installation or a server-to-server installation of the IBM MQ server:
- A local server installation has no communication links with other IBM MQ installations.
- A server-to-server installation does have links to other installations.

You can also verify that your IBM MQ MQI client installation completed successfully and that the communication link is working.

Procedure

- To verify a local server installation, see "Verifying a local server installation on Windows."
- To verify a server-to-server installation, see "Verifying a server-to-server installation on Windows" on page 500.
- To verify a client installation, see "Verifying a client installation on Windows" on page 505.

Verifying a local server installation on Windows

Windows

You can use either the command line or the postcard application to verify a local (stand-alone) installation on Windows.

About this task

You can use the command line to verify that IBM MQ is successfully installed, and that the associated communication links are working properly.

You can also verify an installation using the postcard application. The postcard application is Java based and requires a system with the ability to view a graphical display.

Procedure

- To use the command line to verify an installation, see "Verifying a local server installation using the command line on Windows" on page 497.
- To use the postcard application to verify an installation, see "Verifying a local server installation using the Postcard application on Windows" on page 498.

Verifying a local server installation using the command line on Windows: ▶ Windows

On Windows systems, you can verify a local installation by using the command line to create a simple configuration of one queue manager and one queue. You can also verify an installation using the postcard application.

Before you begin

To verify the installation, you must first install the samples package.

Before beginning the verification procedure, you might want to check that you have the latest fixes for your system. For more information about where to find the latest updates, see "Checking requirements on Windows" on page 444.

About this task

Use the following steps to configure your default queue manager from the command line. After the queue manager is configured, use the `amqsput` sample program to put a message on the queue. You then use the `amqsget` sample program to get the message back from the queue.

IBM MQ object definitions are case-sensitive. Any text entered as an MQSC command in lowercase is converted automatically to uppercase unless you enclose it in single quotation marks. Make sure that you type the examples exactly as shown.

Procedure

1. Set up your environment:
   a. Set up environment variables for use with a particular installation by entering the following command:
      ```
      MQ_INSTALLATION_PATH\bin\setmqenv -s
      ```
      where *MQ_INSTALLATION_PATH* refers to the location where IBM MQ is installed.
   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.
2. Create a queue manager called `QMA` by entering the following command:
   ```
   crtmqm QMA
   ```
   Messages indicate when the queue manager is created, and when the default IBM MQ objects are created.
3. Start the queue manager by entering the following command:
   ```
   strmqm QMA
   ```
   A message indicates when the queue manager starts.
4. Start MQSC by entering the following command:
   ```
   runmqsc QMA
   ```
   A message indicates when MQSC starts. MQSC has no command prompt.
5. Define a local queue called `QUEUE1` by entering the following command:
   ```
   DEFINE QLOCAL (QUEUE1)
   ```
   A message indicates when the queue is created.

6. Stop MQSC by entering the following command:

```
end
```

Messages are shown, followed by the command prompt.

Note: Subsequent steps require that the samples package is installed.

7. Put a message on the queue by entering the following command:

```
amqsput QUEUE1 QMA
```

The following messages are shown:

```
Sample AMQSPUT0 start
target queue is QUEUE1
```

8. Type some message text on one or more lines, where each line is a different message. 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. Get the messages from the queue, by entering the following command:

```
amqsget QUEUE1 QMA
```

The sample program starts, and your messages are displayed.

Results

You have successfully verified your local installation.

Verifying a local server installation using the Postcard application on Windows: ▶ Windows

Sending messages successfully between two Postcard applications verifies a local installation.

Before you begin

The postcard application is Java based and requires a system with the ability to view a graphical display.

You must ensure that you are a member of the IBM MQ administrators group (**mqm**).

Note: Using Postcard to verify an IBM MQ installation is only possible if there is one IBM MQ installation on that box. The Default Configuration wizard will not create a default configuration if a queue manager already exists on the box. The Default Configuration wizard will run on any installation on a box but only one default configuration can be created per box. Using Postcard to verify second and subsequent installations of IBM MQ on the same box is not possible.

To verify that the local installation is working, you can run two instances of the Postcard application on the same server. The postcard application can send messages to, and receive messages from, other postcard applications. Successful sending and receiving of messages verifies that IBM MQ is installed and working correctly on the server.

Procedure

1. Log on as a user in group **mqm**.
2. Start the postcard application in one of the following ways:
   a. From the command line:
      1) Change the directory to *MQ_INSTALLATION_PATH*\java\bin. *MQ_INSTALLATION_PATH* represents the high-level directory in which IBM MQ is installed.

   2) Run the postcard application by entering the following command:
      ```
      postcard
      ```
   b. From the IBM MQ Explorer: On Windows systems, you can start IBM MQ Explorer by using the system menu, the **`MQExplorer`** command (preferred command), or the `MQExplorer` executable file. The **`strmqcfg`** command is still usable.
      1) If the Welcome to IBM MQ Explorer Content view page does not show, click **IBM MQ** in the Navigator view to show the Welcome page.
      2) Click **Launch Postcard** to start the Postcard.
3. At the Postcard - Sign On window, type in a nickname to use to send messages within the Postcard application (for example, `User1`).
4. Select the queue manager to use as the mailbox:
   - If you do not have any queue managers, you are prompted to either launch the Default Configuration or close the Postcard application. Launching the Default Configuration creates a default queue manager.
   - If the only queue manager on your server is the default queue manager, this queue manager is used automatically for the postcard application. The default queue manager is created by running the Default Configuration wizard
   - If you have created your own queue managers, but you have not run the Default Configuration wizard, select an appropriate queue manager from the list.
   - If you have run the Default Configuration wizard and you want to use the default queue manager, but there are other queue managers on your server, select the **Advanced** check box. Then select **Use Default Configuration as mailbox**.
   - If you have run the Default Configuration wizard and also created your own queue managers, and you do not want to use the default queue manager, select the **Advanced** check box. Then select **Choose queue manager as mailbox**, and then select the appropriate queue manager from the list.

   When your selection is complete, click **OK** to display your first Postcard window.
5. Run a second instance of the Postcard application by following the steps used to open the first instance of the Postcard application.
6. The Postcard - Sign On panel is displayed again. Type in a second nickname to use to send messages within this second Postcard application (for example, `User2`).
7. Repeat the selection of the queue manager that you want to use as the mailbox (as described in step 4). The queue manager you select for this second Postcard must be the same queue manager as used for the first instance of the Postcard application.
8. In the first Postcard, (User1), enter the nickname (`User2`) for the second Postcard application in the **To:** field. Because the sender and receiver are on the same server, you can leave the **On:** field blank.
9. Type a message in the **Message:** field and click **Send**.
10. The **Postcards sent and received** area of the Postcard shows details of the message. In the sending Postcard, the message is displayed as sent. In the receiving Postcard, the message is displayed as received.
11. In the receiving Postcard, (User2), double-click the message in the **Postcards sent and received** area to view it. When this message arrives, it verifies that IBM MQ is correctly installed.

What to do next

Depending on your situation, you might want to do the following tasks:
- Install IBM MQ on other servers. Follow the installation procedure for the appropriate platform. Ensure that you use the **Join Default Cluster** window in the Default Configuration wizard to add the other servers to the cluster on your first server.
- Install the IBM MQ MQI client on other servers.

- Continue with further administration tasks, see Administering IBM MQ.

Verifying a server-to-server installation on Windows

Windows

You can use either the command line or the postcard application to verify a server-to-server installation on Windows.

Before you begin

For a server-to-server verification, the communication links between the two systems must be checked. Before you can do the verification, you must therefore ensure that the communications protocol is installed and configured on both systems.

On Windows, IBM MQ supports TCP, SNA, NetBios, and SPX.

The examples in this task use TCP/IP. If you do not use TCP, see Setting up communication for Windows.

About this task

For a server-to server installation, you can use the command line to verify that IBM MQ is successfully installed, and that the associated communication links are working properly.

You can also verify an installation using the postcard application. The postcard application is Java based and requires a system with the ability to view a graphical display.

Procedure

- To use the command line to verify an installation, see "Verifying a server-to-server installation using the command line on Windows."
- To use the postcard application to verify an installation, see "Verifying a server-to-server installation using the Postcard application on Windows" on page 503.

Verifying a server-to-server installation using the command line on Windows: Windows

You can verify a server-to-server installation using two servers, one as a sender and one as a receiver.

Before you begin

- Make sure that TCP/IP and IBM MQ are installed on both servers (see "Verifying a server-to-server installation on Windows").
- Make sure that you are a member of the IBM MQ administrators group (**`mqm`**) on each server.
- Decide which installation is the sender server and which installation is the receiver server. The installations might be on the same system, or on different systems.

About this task

IBM MQ object definitions are case-sensitive. Any text entered as an MQSC command in lowercase is converted automatically to uppercase unless you enclose it in single quotation marks. Make sure that you type the examples exactly as shown.

Procedure

1. On the **`receiver`** server:
   a. Check which ports are free, for example by running **`netstat`**. For more information about this command, see the documentation of your operating system.

If port 1414 is not in use, make a note of 1414 to use as the port number in step 2 g. Use the same number for the port for your listener later in the verification. If it is in use, note a port that is not in use; for example 1415.

b. Set up the environment for the installation you are using by entering the following command at the command prompt:

   ```
   MQ_INSTALLATION_PATH\bin\setmqenv -s
   ```

   where *MQ_INSTALLATION_PATH* refers to the location where IBM MQ is installed.

c. Create a queue manager called `QMB` by entering the following command at the command prompt:

   ```
   crtmqm QMB
   ```

   Messages tell you that the queue manager has been created, and that the default IBM MQ objects have been created.

d. Start the queue manager by entering the following command:

   ```
   strmqm QMB
   ```

   A message tells you when the queue manager has started.

e. Start MQSC by entering the following command:

   ```
   runmqsc QMB
   ```

   A message tells you that MQSC has started. MQSC has no command prompt.

f. Define a local queue called `RECEIVER.Q` by entering the following command:

   ```
   DEFINE QLOCAL (RECEIVER.Q)
   ```

   A message tells you the queue has been created.

g. Define a listener by entering the following command:

```
DEFINE LISTENER (LISTENER1) TRPTYPE (TCP) CONTROL (QMGR) PORT ( PORT_NUMBER )
```

   Where *port_number* is the name of the port the listener runs on. This number must be the same as the number used when defining your sender channel.

h. Start the listener by entering the following command:

   ```
   START LISTENER (LISTENER1)
   ```

   Note: Do not start the listener in the background from any shell that automatically lowers the priority of background processes.

i. Define a receiver channel by entering the following command:

   ```
   DEFINE CHANNEL (QMA.QMB) CHLTYPE (RCVR) TRPTYPE (TCP)
   ```

   A message tells you when the channel has been created.

j. End MQSC by typing:

   ```
   end
   ```

   Some messages are displayed, followed by the command prompt.

2. On the **sender** server:

   a. Set up the environment for the installation you are using by entering the following command at the command prompt:

      ```
      MQ_INSTALLATION_PATH\bin\setmqenv -s
      ```

      where *MQ_INSTALLATION_PATH* refers to the location where IBM MQ is installed.

   b. Create a queue manager called `QMA` by entering the following command at the command prompt:

      ```
      crtmqm QMA
      ```

Messages tell you that the queue manager has been created, and that the default IBM MQ objects have been created.

c. Start the queue manager, by entering the following command:

```
strmqm QMA
```

A message tells you when the queue manager has started.

d. Start MQSC by entering the following command:

```
runmqsc QMA
```

A message tells you that an MQSC session has started. MQSC had no command prompt.

e. Define a local queue called `QMB` (to be used as a transmission queue) by entering the following command:

```
DEFINE QLOCAL (QMB) USAGE (XMITQ)
```

A message tells you when the queue has been created.

f. Define a local definition of the remote queue by entering the following command:

```
DEFINE QREMOTE (LOCAL.DEF.OF.REMOTE.QUEUE) RNAME (RECEIVER.Q) RQMNAME ('QMB') XMITQ (QMB)
```

g. Define a sender channel by entering the following command:

```
DEFINE CHANNEL (QMA.QMB) CHLTYPE (SDR) CONNAME ('CON-NAME(PORT)') XMITQ (QMB) TRPTYPE (TCP)
```

con-name is the TCP/IP address of the receiver system. If both installations are on the same system, the *con-name* is `localhost`. *port* is the port you noted in 1 a. If you do not specify a port, the default value of 1414 is used.

h. Start the sender channel by entering the following command:

```
START CHANNEL(QMA.QMB)
```

The receiver channel on the receiver server starts automatically when the sender channel starts.

i. Stop MQSC by entering the following command:

```
end
```

Some messages are displayed, followed by the command prompt.

j. If both the sender server and receiver server are installations on the same system, check that the queue managers have been created on different installations by entering the following command:

```
dspmq -o installation
```

If the queue managers are on the same installation, move either QMA to the sender installation or QMB to the receiver installation by using the **setmqm** command. For more information, see setmqm.

k. Put a message on the local definition of the remote queue, which in turn specifies the name of the remote queue. Enter the following command:

```
amqsput LOCAL.DEF.OF.REMOTE.QUEUE QMA
```

A message tells you that `amqsput` has started.

l. Type some message text on one or more lines, followed by a blank line. A message tells you that `amqsput` has ended. Your message is now on the queue and the command prompt is displayed again.

3. On the **receiver** server:

a. Get the message from the queue on the receiver by entering the following command:

```
amqsget RECEIVER.Q QMB
```

The sample program starts, and your message is displayed. After a pause, the sample ends. Then the command prompt is displayed.

Results

You have now successfully verified the server-to-server installation.

Verifying a server-to-server installation using the Postcard application on Windows: Windows

You can use two instances of the Postcard application to verify that a server-to-server installation is working.

Before you begin

You can use the Postcard application on two servers, one instance of the Postcard application on each server, to verify that a server-to-server installation is working. Successful sending and receiving of messages verifies that IBM MQ is successfully installed, and that communication between the two servers is working correctly.

Note:

- If the system has multiple IBM MQ installations, ensure that Postcard has not been run before on any installations on that server. As the default configuration can only exist on one IBM MQ MQ installation per system, the Default Configuration wizard and Postcard can not be used for verification of a second or any subsequent installation.
- The two server installations must be on different systems to do a server-to-server verification using the postcard application. To verify a server-to-server installation on the same machine, you can use the command line.
- Make sure that TCP/IP and IBM MQ are installed on both servers.
- Make sure that your systems are able to view a graphical display.
- Make sure that you are a member of the IBM MQ administrators group (**mqm**) on each server.
- Check that one of the following scenarios applies:
  - Neither server has had any queue managers created.
  - Use the Default Configuration wizard to create default queue managers on each server and link them to the default cluster.

    Details on how to use the Default Configuration wizard are provided in this topic.
  - Both servers have existing queue managers and these queue managers are in the same cluster.

    If your queue managers are not in the same cluster, create new queue managers on both servers. Then create a cluster, and ensure that the queue managers that you create on each server belong to that cluster.
  - You have configured channels to communicate between the two servers.

    For instructions on how to set up the channels, see "Verifying a server-to-server installation using the command line on Windows" on page 500. After you have set up the channels, follow the instructions in this topic to verify your server-to-server installation.

Procedure

1. On the first server, log on as a user in group **mqm**.
2. Start the postcard application in one of the following ways:
   a. From the command line:
      1) Change the directory to *MQ_INSTALLATION_PATH*`\java\bin`. *MQ_INSTALLATION_PATH* represents the high-level directory in which IBM MQ is installed.
      2) Run the postcard application by entering the following command:
      ```
      postcard
      ```
   b. From the IBM MQ Explorer: On Windows systems, you can start IBM MQ Explorer by using the system menu, the `MQExplorer` executable file, or the **strmqcfg** command.

1) If the Welcome to IBM MQ Explorer Content view page does not show, click **IBM MQ** in the Navigator view to show the Welcome page.
2) Click **Launch Postcard** to start the Postcard.

3. At the Postcard - Sign On window, type a nickname to use to send messages within the Postcard application. For example, `User1` for the first server, and `User2` for the second server.
4. Select the queue manager to use as the mailbox:
   - If you do not have any queue managers, you are prompted to either launch the Default Configuration or close the Postcard application. Work through the Default Configuration wizard. When you get to the option to join the queue manager to the default cluster, tick the check box. On the next screen:
     - For the first server, select **yes, make it the repository for the cluster**.
     - For the second server, select **No another computer has already joined the cluster as a repository**. When requested, enter the location of the repository, by typing the name of the sender server.
   - If the only queue manager on your server is the default queue manager, this queue manager is used automatically for the postcard application. The default queue manager is created by running the Default Configuration wizard
   - If you have created your own queue managers, but you have not run the Default Configuration wizard, select an appropriate queue manager from the list.
   - If you have run the Default Configuration wizard and you want to use the default queue manager, but there are other queue managers on your server, select the **Advanced** check box. Then select **Use Default Configuration as mailbox**.
   - If you have run the Default Configuration wizard and also created your own queue managers, and you do not want to use the default queue manager, select the **Advanced** check box. Then select **Choose queue manager as mailbox**, and then select the appropriate queue manager from the list.

   When your selection is complete, click **OK**.
5. Select the queue manager to use as the mailbox:
   - If you do not have any queue managers, you are prompted to either launch the Default Configuration or close the Postcard application. Work through the Default Configuration wizard. When you get to the option to join the queue manager to the default cluster, tick the check box. On the next screen:
     - For the first server, select **yes, make it the repository for the cluster**.
     - For the second server, select **No another computer has already joined the cluster as a repository**. When requested, enter the location of the repository, by typing the name of the sender server.
   - If the only queue manager on your server is the default queue manager, this queue manager is used automatically for the postcard application. The default queue manager is created by running the Default Configuration wizard
   - If you have created your own queue managers, but you have not run the Default Configuration wizard, select an appropriate queue manager from the list.
   - If you have run the Default Configuration wizard and you want to use the default queue manager, but there are other queue managers on your server, select the **Advanced** check box. Then select **Use Default Configuration as mailbox**.
   - If you have run the Default Configuration wizard and also created your own queue managers, and you do not want to use the default queue manager, select the **Advanced** check box. Then select **Choose queue manager as mailbox**, and then select the appropriate queue manager from the list.

   When your selection is complete, click **OK**.
6. Complete steps 1 - 5 for the second server.
7. In the Postcard on the first server:
   a. Enter the nickname (`user2`) for the Postcard application on the second server in the **To:** field.

   b. Enter the queue manager on the second server in the **On:** field.
   c. Type a message in the **Message:** field and click **Send**.
8. In the Postcard on the second server:
   a. In the **Postcards sent and received**, double-click the message marked as received to view the message from the first server.
   b. Optional: Send a postcard to the first server by adapting the instructions in step 7. You must enter details of the first server in the **To:** field and the **On:** field.

   The messages verify that IBM MQ is correctly installed and that your communication link between the two servers is working correctly.

Verifying a client installation on Windows

Windows

You can verify that your IBM MQ MQI client installation completed successfully and that the communication link is working.

About this task

The verification procedure shows how to create a queue manager called `queue.manager.1`, a local queue called `QUEUE1`, and a server-connection channel called `CHANNEL1` on the server.

It shows how to create the client-connection channel on the IBM MQ MQI client workstation. It then shows how to use the sample programs to put a message onto a queue, and get the message from the queue.

The example does not address any client security issues. See Setting up IBM MQ MQI client security for details if you are concerned with IBM MQ MQI client security issues.

The verification procedure assumes that:
- The full IBM MQ server product has been installed on a server.
- The server installation is accessible on your network.
- The IBM MQ MQI client software has been installed on a client system.
- The IBM MQ sample programs have been installed.
- TCP/IP has been configured on the server and client systems. For more information, see Configuring connections between the server and client.

Procedure

1. Set up the server and client:
   - To set up the server and client by using the command line, follow the instructions in “Setting up the server and client using the command line on Windows” on page 506.
   - To set up the server and client by using IBM MQ Explorer, follow the instructions in “Setting up the server and client using IBM MQ Explorer on Windows” on page 509.
2. Test the communications between client and server, using the instructions in “Testing communication between a client and a server on Windows” on page 512.

Related tasks:

"Installing an IBM MQ client on Windows" on page 481
This topic describes how to install IBM MQ client on Windows systems. This procedure can be used for installing a first or a subsequent installation.

Setting up the server and client using the command line on Windows: Windows

You can use the command line to create the objects that you need to use to verify a client installation on Linux. On the server you create a queue manager, a local queue, a listener, and a server-connection channel. You must also apply security rules to allow the client to connect and make use of the queue defined. On the client you create a client-connection channel. After setting up the server and client, you can then use the sample programs to complete the verification procedure.

Before you begin

Before starting this task, review the information in "Verifying a client installation on Windows" on page 505.

About this task

This task explains how to use the command line to set up the server and client so that you can verify your client installation.

If you prefer to use IBM MQ Explorer, see "Setting up the server and client using IBM MQ Explorer on Windows" on page 509.

Procedure

1. Set up the server by following the instructions in "Setting up the server using the command line on Windows."
2. Set up the client by following instructions in "Connecting to a queue manager, using the `MQSERVER` environment variable on Windows" on page 508.

What to do next

Test the communications between client and server by following the instructions in "Testing communication between a client and a server on Windows" on page 512.

Setting up the server using the command line on Windows: Windows

Follow these instructions to create a queue manager, queue, and channel on the server. You can then use these objects to verify the installation.

About this task

These instructions assume that no queue manager or other IBM MQ objects have been defined.

IBM MQ object definitions are case-sensitive. Any text entered as an MQSC command in lowercase is converted automatically to uppercase unless you enclose it in single quotation marks. Make sure that you type the examples exactly as shown.

Procedure

1. Create a user ID on the server that is not in the `mqm` group. This user ID must exist on the server and client. This is the user ID that the sample applications must be run as, otherwise a 2035 error is returned.

2. You must set various environment variables so that the installation can be used in the current shell. You can set the environment variables by entering the following command:

```
MQ_INSTALLATION_PATH\bin\setmqenv -s
```

   where *MQ_INSTALLATION_PATH* refers to the location where IBM MQ is installed
3. Create a queue manager called QUEUE.MANAGER.1 by entering the following command:

   ```
   crtmqm QUEUE.MANAGER.1
   ```

   You see messages telling you that the queue manager has been created.
4. Start the queue manager by entering the following command:

   ```
   strmqm QUEUE.MANAGER.1
   ```

   A message tells you when the queue manager has started.
5. Start MQSC by entering the following command:

   ```
   runmqsc QUEUE.MANAGER.1
   ```

   A message tells you that an MQSC session has started. MQSC has no command prompt.
6. Define a local queue called QUEUE1 by entering the following command:

   ```
   DEFINE QLOCAL(QUEUE1)
   ```

   A message tells you when the queue has been created.
7. Allow the user ID that you created in step 1 to use QUEUE1 by entering the following command:

   ```
   SET AUTHREC PROFILE(QUEUE1) OBJTYPE(QUEUE) PRINCIPAL(' non_mqm_user ') AUTHADD(PUT,GET)
   ```

   where *non_mqm_user* is the user ID created in step 1. A message tells you when the authorization has been set. You must also run the following command to give the user ID authority to connect:

   ```
   SET AUTHREC OBJTYPE(QMGR) PRINCIPAL(' non_mqm_user ') AUTHADD(CONNECT)
   ```

   If this command is not run, a 2305 stop error is returned.
8. Define a server-connection channel by entering the following command:

   ```
   DEFINE CHANNEL (CHANNEL1) CHLTYPE (SVRCONN) TRPTYPE (TCP)
   ```

   A message tells you when the channel has been created.
9. Allow your client channel to connect to the queue manager and run under the user ID that you created in step 1, by entering the following MQSC command:

   ```
   SET CHLAUTH(CHANNEL1) TYPE(ADDRESSMAP) ADDRESS(' client_ipaddr ') MCAUSER(' non_mqm_user ')
   ```

   where *client_ipaddr* is the IP address of the client system, and *non_mqm_user* is the user ID created in step 1. A message tells you when the rule has been set.
10. Define a listener by entering the following command:

    ```
    DEFINE LISTENER (LISTENER1) TRPTYPE (TCP) CONTROL (QMGR) PORT (port_number)
    ```

    where *port_number* is the number of the port the listener is to run on. This number must be the same as the number used when defining your client-connection channel in "Installing an IBM MQ client on Windows" on page 481.

    Note: If you omit the port parameter from the command, a default value of 1414 is used for the listener port. If you want to specify a port other than 1414, you must include the port parameter in the command, as shown.
11. Start the listener by entering the following command:

    ```
    START LISTENER (LISTENER1)
    ```
12. Stop MQSC by entering:

```
end
```

You see some messages, followed by the command prompt.

What to do next

Follow the instructions to set up the client. See "Connecting to a queue manager, using the `MQSERVER` environment variable on Windows."

Connecting to a queue manager, using the `MQSERVER` environment variable on Windows: Windows

When an IBM MQ application is run on the IBM MQ MQI client, it requires the name of the MQI channel, the communication type, and the address of the server to be used. Provide these parameters by defining the `MQSERVER` environment variable.

Before you begin

Before you start this task, you must complete the task, "Setting up the server using the command line on Windows" on page 506, and save the following information:
- The host name or IP address of the server and port number that you specified when creating the listener.
- The channel name of the server-connection channel.

About this task

This task describes how to connect an IBM MQ MQI client, by defining the `MQSERVER` environment variable on the client.

You can give the client access to the generated client channel definition table, `amqclchl.tab` instead; see Accessing client-connection channel definitions.

Alternatively, on Windows, if Active Directory support is enabled, the client discovers the client-connection information dynamically from the Active Directory.

Procedure
1. Log in as the userid that you created in Step 1 of "Setting up the server using the command line on Windows" on page 506.
2. Check the TCP/IP connection. From the client, enter one of the following commands:
   - `ping server-hostname`
   - `ping n.n.n.n`

     `n.n.n.n` represents the network address. You can set the network address in IPv4 dotted decimal form, for example, `192.0.2.0`. Alternatively, set the address in IPv6 hexadecimal form, for example `2001:0DB8:0204:acff:fe97:2c34:fde0:3485`.

   If the **`ping`** command fails, correct your TCP/IP configuration.
3. Set the `MQSERVER` environment variable. From the client, enter the following command:
   ```
   SET MQSERVER=CHANNEL1/TCP/server-address (port)
   ```
   Where:
   - *CHANNEL1* is the server-connection channel name.
   - *server-address* is the TCP/IP host name of the server.
   - *port* is the TCP/IP port number the server is listening on.

If you do not give a port number, IBM MQ uses the one specified in the `qm.ini` file, or the client configuration file. If no value is specified in these files, IBM MQ uses the port number identified in the TCP/IP services file for the service name `MQSeries`. If an `MQSeries` entry in the services file does not exist, a default value of `1414` is used. It is important that the port number used by the client and the port number used by the server listener program are the same.

What to do next

Use the sample programs to test communication between the client and server; see "Testing communication between a client and a server on Windows" on page 512.

Setting up the server and client using IBM MQ Explorer on Windows: Windows

You can use IBM MQ Explorer to create the objects that you need to use to verify a client installation on Windows. On the server, you create a queue manager, a local queue, a listener and a server-connection channel. On the client system you create a client-connection channel. Then from the command line you use the sample PUT and GET programs to complete the verification procedure.

Before you begin

Before starting this task, review the information in "Verifying a client installation on Windows" on page 505.

About this task

This task explains how to use IBM MQ Explorer to set up the server and client so that you can verify your client installation.

If you prefer to use the command line, see "Setting up the server and client using the command line on Windows" on page 506.

Procedure

1. Set up the server by following the instructions in "Setting up the server using IBM MQ Explorer on Windows" on page 510.
2. Set up the client by following instructions in "Setting up the client using IBM MQ Explorer on Windows" on page 511.

What to do next

Test the communications between client and server by following the instructions in "Testing communication between a client and a server on Windows" on page 512.

Related tasks:

"Installing an IBM MQ client on Windows" on page 481
This topic describes how to install IBM MQ client on Windows systems. This procedure can be used for installing a first or a subsequent installation.

Setting up the server using IBM MQ Explorer on Windows: ▶ Windows

On the server, you create a queue manager, a local queue, a listener and a server-connection channel. On the client system you create a client-connection channel. Then from the command line you use the sample PUT and GET programs to complete the verification procedure.

About this task

You can use the IBM MQ Explorer to create a queue manager, queue and server-connection channel on Windows. This topic describes the tasks necessary to set up the server.

Procedure

1. Create a queue manager:
   a. Open IBM MQ Explorer.
   b. Right-click the folder called **Queue Managers**, select **New** > **Queue Manager**.
   c. In the first entry field, type the queue manager name, *QUEUE.MANAGER.1*, and click **Finish**.
2. Create a local queue:
   a. Expand the queue manager you have just created and right-click **queues**.
   b. Select **New** > **Local Queue**.
   c. Enter the queue name, *QUEUE1*, and click **Finish**.
3. Define the server-connection channel:
   a. Right-click **Channels**.
   b. Select **New** > **Server Connection Channel**.
   c. Enter the channel name, *CHANNEL1*, and click **Next**.
   d. In the dialog navigation pane, click **MCA** to open the MCA page.
   e. In the MCA User ID field, enter a userid that is a member of the mqm group, typically your own.
   f. Click **Finish**.
4. Run the listener.

   The listener is automatically started when the queue manager is configured. To check that the listener is running, open **Listeners** and look for `LISTENER.TCP`.

What to do next

Set up the client. See "Setting up the client using IBM MQ Explorer on Windows" on page 511.

Related tasks:

"Testing communication between a client and a server on Windows" on page 512
On the IBM MQ MQI client workstation, use the amqsputc sample program to put a message on the queue at the server workstation. Use the amqsgetc sample program to get the message from the queue back to the client.

"Installing an IBM MQ client on Windows" on page 481
This topic describes how to install IBM MQ client on Windows systems. This procedure can be used for installing a first or a subsequent installation.

Setting up the client using IBM MQ Explorer on Windows: Windows

You can use IBM MQ Explorer to define the client-connection if you are setting up the client and server on the same workstation on a Windows system.

Procedure

1. Select the queue manager, *QUEUE.MANAGER.1*
2. Open the **Channels** folder, then right-click **Client Connections** > **New** > **Client-connection Channel...**
3. Enter the channel name, *CHANNEL1*, for the client connection, and click **Next**.
4. Enter the queue manager name, *QUEUE.MANAGER.1*
5. Enter the following string as the connection name:
   ```
   server-address (port)
   ```
   Where:
   - `server-address` is the TCP/IP host name of the server
   - `port` is the TCP/IP port number the server is listening on
6. Click Finish.
7. From the command line, set the `MQCHLLIB` environment variable. Enter the following command:
   ```
   SET MQCHLLIB= MQ_INSTALLATION_PATH\qmgrs\QUEUE!MANAGER!1\@ipcc
   ```
   where `MQ_INSTALLATION_PATH` represents the high-level directory in which IBM MQ is installed

   Note: The queue manager name contains "**.**". IBM MQ creates the queue manager directory with the name, `QUEUE!MANAGER!1`

What to do next

Use the sample programs to test communication between the client and server. See "Testing communication between a client and a server on Windows" on page 512.

Related tasks:

"Setting up the server and client using IBM MQ Explorer on Windows" on page 509
You can use IBM MQ Explorer to create the objects that you need to use to verify a client installation on Windows. On the server, you create a queue manager, a local queue, a listener and a server-connection channel. On the client system you create a client-connection channel. Then from the command line you use the sample PUT and GET programs to complete the verification procedure.

"Installing an IBM MQ client on Windows" on page 481
This topic describes how to install IBM MQ client on Windows systems. This procedure can be used for installing a first or a subsequent installation.

Testing communication between a client and a server on Windows: Windows

On the IBM MQ MQI client workstation, use the amqsputc sample program to put a message on the queue at the server workstation. Use the amqsgetc sample program to get the message from the queue back to the client.

Before you begin

Complete the previous topics in this section:
- Set up a queue manager, channels, and queue.
- Open a command window.
- Set system environment variables.

About this task

Note that IBM MQ object definitions are case-sensitive. Text entered as an MQSC command in lowercase is converted automatically to uppercase unless you enclose it in single quotation marks. Make sure that you type the examples exactly as shown.

Procedure

1. Change into the *MQ_INSTALLATION_PATH*\Tools\C\Samples\Bin directory for 32 bit systems or the *MQ_INSTALLATION_PATH*\Tools\C\Samples\Bin64 directory for 64 bit systems. *MQ_INSTALLATION_PATH* represents the high-level directory in which IBM MQ is installed.
2. You must set certain environment variables so that the installation can be used in the current shell. You can set the environment variables by entering the following command:
   ```
   MQ_INSTALLATION_PATH\bin\setmqenv -s
   ```
   where *MQ_INSTALLATION_PATH* refers to the location where IBM MQ is installed
3. Start the PUT program for QUEUE1 on QUEUE.MANAGER.1 by entering the following command:
   ```
   amqsputc QUEUE1 QUEUE.MANAGER.1
   ```
   If the command is successful, the following messages are displayed:
   ```
   Sample AMQSPUT0 start target queue is QUEUE1
   ```
   Tip: You might get the error, MQRC_NOT_AUTHORIZED (2035). By default, channel authentication is enabled when a queue manager is created. Channel authentication prevents privileged users accessing a queue manager as an IBM MQ MQI client. For verifying the installation, you can either change the MCA user ID to a non-privileged user, or disable channel authentication. To disable channel authentication run the following MQSC command:
   ```
   ALTER QMGR CHLAUTH(DISABLED)
   ```
   When you finish the test, if you do not delete the queue manager, re-enable channel authentication:
   ```
   ALTER QMGR CHLAUTH(ENABLED)
   ```

4. Type some message text, then press **Enter** twice. The following message is displayed:
   ```
   Sample AMQSPUT0 end
   ```
   Your message is now on the queue that is on the server queue manager.
5. Start the GET program for QUEUE1 on QUEUE.MANAGER.1 by entering the following command:
   ```
   amqsgetc QUEUE1 QUEUE.MANAGER.1
   ```
   The sample program starts, and your message is displayed. After a short pause (approximately 30 seconds), the sample ends and the command prompt is displayed again.

Results

You have now successfully verified the client installation.

What to do next

1. You must set various environment variables on the server so that the installation can be used in the current shell. You can set the environment variables by entering the following command:
   ```
     MQ_INSTALLATION_PATH\bin\setmqenv -s
   ```
   where *MQ_INSTALLATION_PATH* refers to the location where IBM MQ is installed.
2. On the server, stop the queue manager by entering the following command:
   ```
   endmqm QUEUE.MANAGER.1
   ```
3. On the server, delete the queue manager by entering the following command:
   ```
   dltmqm QUEUE.MANAGER.1
   ```

Verifying the installation of MQ Telemetry

There are three ways to verify the installation of MQ Telemetry. Any can be used, regardless of whether MQ Telemetry was installed as a custom installation of IBM MQ, or added to an existing installation of IBM MQ.

About this task

Within IBM MQ you can verify the installation of MQ Telemetry either by using IBM MQ Explorer, or by using the command line.

You can also verify the installation by using the MQTT messaging client for JavaScript in a browser that supports the RFC 6455 (WebSocket) standard. A version of this client is installed with MQ Telemetry, and the latest version is available as part of the IBM Messaging Telemetry Clients SupportPac. To verify the MQ Telemetry installation you do not need the latest version of the client, so you should omit step 1.

Procedure

Verify your installation in one of the following ways:
- By using IBM MQ Explorer as described in “Verifying the installation of MQ Telemetry by using IBM MQ Explorer” on page 540.
- By using the command line as described in “Verifying the installation of MQ Telemetry using the command line” on page 542.

Uninstalling IBM MQ on Windows

Windows

You can uninstall the IBM MQ MQI clients and servers on Windows systems by using the control panel, the command line (**msiexec**), **MQParms**, or by using the installation media, in which case you can optionally remove queue managers as well.

Before you begin

By default, uninstallation logging is not enabled on Windows. To ensure that you receive an uninstallation log, carry out the following procedure:

1. In a command prompt, open the registry editor by issuing the command **regedit**.
2. Create, or edit, the appropriate registry key: `HKEY_LOCAL_MACHINE\Software\Policies\Microsoft\Windows\Installer`
3. Under this registry key add the following information:

   Name Logging

   Data type
   REG_SZ

   Value voicewarmup
4. Save the updated registry key.

Procedure

The first part of the procedure ensures that there are no IBM MQ programs or processes running:

1. If you are running IBM MQ with the Microsoft Cluster Service (MSCS), remove the queue managers from MSCS control before uninstalling IBM MQ. Perform the following steps for each queue manager currently under MSCS control :
   a. Take the queue manager resource offline.
   b. Destroy the resource instance.
   c. Migrate the queue manager files back from shared drives. This step is shown as optional in Removing a queue manager from MSCS control. However, it is mandatory in this case.
2. Stop all IBM MQ applications associated with the installation you are uninstalling.
3. Close all Managed File Transfer agents. If you have a Managed File Transfer Agent running, close it by using the **fteStopAgent** command; see fteStopAgent (stop a Managed File Transfer Agent).
4. For a server installation, end all IBM MQ activity:
   a. Log in as a user in the group mqm.
   b. Stop all running queue managers and listeners by using the IBM MQ Explorer, or by entering the following commands:
      1) Set up your environment to work with the installation you want to uninstall by entering the following command:
      ```
      MQ_INSTALLATION_PATH\bin\setmqenv -s
      ```
      where *MQ_INSTALLATION_PATH* is the location where IBM MQ is installed.
      2) For each queue manager, enter the following command to stop the queue manager:
      ```
      endmqm queue_manager_name
      ```
      3) For each queue manager, enter the following command to stop any listeners associated with the queue manager:
      ```
      endmqlsr -m queue_manager_name
      ```
5. Stop IBM MQ. To do this right-click the **IBM MQ** icon in the system tray, then select **Stop IBM MQ**.

6. Close all IBM MQ windows.
7. Stop any monitoring service.

When all processes associated with IBM MQ are no longer running, you can uninstall IBM MQ:

8. Uninstall IBM MQ by using one of the following methods:
   - Use the Windows Control Panel. This process is described in: "Uninstalling IBM MQ using the control panel" on page 516. This method does not remove the queue manager data.
   - Use the command line by running the **msiexec** command as described in: "Uninstalling IBM MQ using the command line" on page 516. This method does not remove the queue manager data.
   - Use the appropriate parameters with **MQParms**. This process is described in "Uninstalling IBM MQ using MQParms" on page 518. This method does not remove the queue manager data.
   - Use the installation media, by selecting the appropriate option as described in: "Uninstalling IBM MQ on Windows using the installation media" on page 518. The option to remove queue manager data is displayed in the Removing Server feature panel, if appropriate.

   If you have to cancel the uninstallation process before it is finished, you might have to reconfigure IBM MQ with the Prepare IBM MQ wizard because the rollback of the deletion of the IBM MQ service is unable to set the service's user account password. Use the following command to reconfigure IBM MQ:

   ```
   MQ_INSTALLATION_PATH\bin\amqmjpse.exe -r
   ```

   For more information about the Prepare IBM MQ wizard, see "Configuring IBM MQ with the Prepare IBM MQ wizard" on page 471.
9. Check the Windows event log and restart the system if necessary. If event ID 10005 is written to the Windows event log, you must restart the system to complete the uninstallation process.
10. Optional: If you are uninstalling the last or only installation of IBM MQ, you can remove all the information about previous installations that is retained on the system, if you want to.

    Two registry values remain after uninstallation:
    - 32 bit systems:
      – `My Computer\HKEY_LOCAL_MACHINE\SOFTWARE\IBM\WebSphere MQ\LogDefaultPath`
      – `My Computer\HKEY_LOCAL_MACHINE\SOFTWARE\IBM\WebSphere MQ\WorkPath`
    - 64 bit systems:
      – `My Computer\HKEY_LOCAL_MACHINE\SOFTWARE\WOW6432Node\IBM\WebSphere MQ\LogDefaultPath`
      – `My Computer\HKEY_LOCAL_MACHINE\SOFTWARE\WOW6432Node\IBM\WebSphere MQ\WorkPath`

    Data folders will also remain and are located at *MQ_DATA_PATH*`\Config`, where *MQ_DATA_PATH* is the location of the IBM MQ data directory. Most of the remaining files contain text such as INI files, error logs, and FDC files. The executable shared library `mqzsd.dll` also remains.

    If a client is installed on a system where the `LogDefaultPath` registry value remains from a previous server installation, a client installation will attempt to create this directory if it does not already exist. If this behavior is not wanted, remove the LogDefaultPath registry value before installing the client.

Uninstalling IBM MQ using the control panel

Windows

You can uninstall IBM MQ by using the control panel to remove all currently installed features.

Before you begin

Start the uninstalling process by following the steps described in "Uninstalling IBM MQ on Windows" on page 514.

If you no longer require the queue managers that are on the system, delete them by using the IBM MQ Explorer or the **dltmqm** command.

Procedure

1. From the Windows taskbar, open the control panel by clicking **Start** > **Settings** > **Control Panel**, or **Start** > **Control Panel**.
2. Open **Programs and Features**.
3. Click **IBM MQ (*installation_name*)**, where *installation_name* is the name of the installation you want to remove.
4. Click **Remove** or **Uninstall** and click **Yes** to confirm. If User Account Control (UAC) is enabled, accept the Windows prompt to allow the uninstallation to run as elevated. The program then begins and runs to completion.

What to do next

Complete the steps that you started in "Uninstalling IBM MQ on Windows" on page 514.

Uninstalling IBM MQ using the command line

Windows

You can uninstall IBM MQ by running the **msiexec** command from the command line to remove all currently installed features.

Before you begin

Start the uninstalling process by following the steps described in "Uninstalling IBM MQ on Windows" on page 514.

If you no longer require the queue managers that are on the system, delete them by using the IBM MQ Explorer or the **dltmqm** command.

About this task

To start uninstalling, use the **msiexec** command.

If you are running IBM MQ on Windows with User Account Control (UAC) enabled, you must invoke the silent uninstallation from an elevated command prompt. Elevate a command prompt by using a right-click to start the command prompt and choose **Run as administrator**.

In all of the examples of commands shown, the variable names used are as follows:

- *installation_name* is the name of the installation you want to remove.
- *product_code* is the value shown for MSIProdCode in the output of the following command:

```
dspmqinst -n installation_name
```

An example of a product code is {0730749B-080D-4A2E-B63D-85CF09AE0EF0}.

- *response_file* is the file that contains the [Response] stanza and the required *property* = *value* pairs. For details about how to create a response file, see “Creating a response file for server installation” on page 463. For details of the parameters you can specify in a response file, see Table 64 on page 458 in “Installing the server using msiexec” on page 455. This is an example of a simple uninstallation [Response] stanza:

  ```
  [Response] REMOVE="ALL"
  ```

Procedure

To uninstall all IBM MQ features, use one of the following methods:

- Run the msiexec command with a parameter that calls a response file.

  A response file is an ASCII text file that contains the parameter values that you want to set for the uninstallation. The response file has a format similar to a Windows .ini file, and contains the stanza [Response]. This stanza contains parameters that the **msiexec** command can use, in the form of *property* = *value* pairs. The **msiexec** command ignores any other stanzas in the file.

  You can set which features to uninstall, and set whether to keep existing queue managers.

  To silently uninstall IBM MQ using a response file, enter the following command:

  ```
  msiexec /x {product_code} /l*v "c:\removal.log" /q USEINI="response_file" INSTALLATIONNAME="installation_name"
  ```

- Enter one of the following commands on the command line:
  - To invoke an interactive uninstallation giving you the option to remove queue manager data (providing there are no other IBM MQ installations remaining):

    ```
    msiexec /x {product_code} /l*v "c:\removal.log" REMOVE="All" INSTALLATIONNAME="installation_name"
    ```

    If you are running IBM MQ on a Windows system with User Account Control (UAC) enabled, you might see Open File - Security Warning dialog boxes during uninstallation that list International Business Machines Limited as the publisher. Click **Run** to allow the uninstallation to continue.
  - To invoke a silent uninstallation that does not remove any queue manager data:

    ```
    msiexec /x {product_code} /l*v "c:\removal.log" /q REMOVE="All" INSTALLATIONNAME="installation_name"
    ```

  - To invoke a silent uninstallation and remove any queue manager data (only valid when removing the final server installation):

    ```
    msiexec /x {product_code} /l*v "c:\removal.log" /q REMOVE="All" KEEPQMDATA="delete"
    INSTALLATIONNAME="installation_name"
    ```

  - To monitor the progress of the uninstalling process and not remove any queue manager data:

    ```
    msiexec /x {product_code} /l*v "c:\removal.log" INSTALLATIONNAME="installation_name"
    ```

    If you are running IBM MQ on a Windows system with User Account Control (UAC) enabled, you might see Open File - Security Warning dialog boxes during uninstallation that list International Business Machines Limited as the publisher. Click **Run** to allow the uninstallation to continue.
  - To invoke a silent uninstallation and not remove any queue manager data:

    ```
    msiexec /x {product_code} /l*v "c:\removal.log" /q INSTALLATIONNAME="installation_name"
    ```

Results

After the command is entered, the command prompt immediately reappears and IBM MQ is uninstalled as a background process. If you entered parameters to produce a log, check this file to see how the uninstallation is progressing. If the uninstallation finishes successfully, you see the message `Removal completed successfully` in the log file.

What to do next

Complete the steps that you started in “Uninstalling IBM MQ on Windows” on page 514.

Uninstalling IBM MQ using MQParms

Windows

You can uninstall IBM MQ by running the **MQParms** command from the command line to remove all currently installed features.

Before you begin

Start the uninstalling process by following the steps described in “Uninstalling IBM MQ on Windows” on page 514.

Procedure

1. Follow the instructions on the MQParms installation pages to uninstall IBM MQ non-interactively. See: “Installing the server using the MQParms command” on page 463.
   a. Set the ADDLOCAL parameter to empty (ADDLOCAL="").
   b. Set the REMOVE parameter to "ALL" (REMOVE="ALL").
2. If you have multiple versions of IBM MQ installed on your system, specify the product code that identifies the installation you want to remove. Type the following command:
   ```
   MQParms.exe parameter_file/i "{product_code}"
   ```
   where
   - *parameter_file* is the file that contains the required parameter values. If this file is not in the same folder as `MQParms.exe`, specify the full path and file name. If you do not specify a parameter file, the default is `MQParms.ini`.
   - *product_code* is the value shown for MSIProdCode in the output of the following command:
     ```
     dspmqinst -n installation_name
     ```
     where *installation_name* is the name of the installation you want to remove. An example of a product code is `{0730749B-080D-4A2E-B63D-85CF09AE0EF0}`.

What to do next

Complete the steps that you started in “Uninstalling IBM MQ on Windows” on page 514.

Uninstalling IBM MQ on Windows using the installation media

Windows

You can uninstall IBM MQ by using the installation media to remove all currently installed features and optionally remove existing queue managers and their data.

Before you begin

Start the uninstalling process by following the steps described in “Uninstalling IBM MQ on Windows” on page 514.

Procedure

1. Insert the IBM MQ for Windows Server DVD into the DVD drive.
2. Start the installation process.
   - If autorun is enabled, the installation process starts automatically.
   - If autorun is not enabled, double-click the **Setup** icon in the root folder of the DVD to start the installation process.

   The IBM MQ Installation Launchpad window opens.

3. Click **IBM MQ Installation**.
4. Click **Launch IBM MQ Installer** and click **Next** until the IBM MQ Program Maintenance panel is displayed with a welcome message. If this panel is not displayed, IBM MQ for Windows is not currently installed.
5. Click **Maintain or upgrade an existing instance** and if there is more than one installation of IBM MQ on the system, select which installation you want to remove. Click **Next** and in the Program Maintenance panel, click **Remove**, then **Next**.
6. If you are uninstalling the last or only server, and there are any queue managers on the system, the Removing Server feature panel is shown. Click one of the following options:
   - **Keep**: keep existing queue managers and their objects.
   - **Remove**: remove existing queue managers and their objects.

   Click **Next**. The Remove IBM MQ panel is displayed, with a summary of the installation to be removed.
7. Click **Remove** to continue. If there are any messages that state that locked files are found, ensure that there are no IBM MQ programs running; see “Uninstalling IBM MQ on Windows” on page 514. When IBM MQ has been uninstalled, a message indicates completion.
8. Click **Finish**.

What to do next

Complete the steps that you started in “Uninstalling IBM MQ on Windows” on page 514.

Installing IBM MQ Advanced for Multiplatforms

Multi MQ Adv.

Installation tasks associated with IBM MQ Advanced for Multiplatforms are grouped in this section.

About this task

IBM MQ Advanced is a single license entitlement that, in addition to IBM MQ itself, provides entitlement to:
- Advanced Message Security
- Managed File Transfer
- MQ Telemetry
- Linux Replicated data queue managers (RDQM)

For more information, see IBM MQ license information.

Procedure
- “Installing and uninstalling AMS on Multiplatforms” on page 520.
- “Installing Managed File Transfer” on page 528.
- “Installing MQ Telemetry” on page 538.
- Linux “Installing RDQM (replicated data queue managers)” on page 545.

Related tasks:

z/OS "Installing IBM MQ Advanced for z/OS" on page 560
Use this topic to understand how you install IBM MQ Advanced for z/OS on your system.

z/OS "Installing IBM MQ Advanced for z/OS, Value Unit Edition" on page 561
Use this topic to understand how you install IBM MQ Advanced for z/OS, Value Unit Edition (VUE) on your system.

Related information:

V 9.0.5 DISPLAY QMGR ADVCAP

V 9.0.5 MQCMD_INQUIRE_Q_MGR MQIA_ADVANCED_CAPABILITY

Installing and uninstalling AMS on Multiplatforms

Multi

Installation and uninstallation, by platform, for Advanced Message Security (AMS) on Multiplatforms.

About this task

Advanced Message Security is a separately installed component of IBM MQ and is another option on the IBM MQ installer. Make sure that you purchase a license for using IBM MQ Advanced before the installation (see IBM MQ license information).

Procedure

- "Installing AMS on Multiplatforms"
- "Uninstalling AMS on Multiplatforms" on page 525

Related tasks:

"Installing Advanced Message Security on z/OS" on page 559
You can install Advanced Message Security component on z/OS by using SMP/E.

Installing AMS on Multiplatforms

Multi

Use the information for your platform to guide you through installing the Advanced Message Security (AMS) component.

Before you begin

Make sure the following IBM MQ components are installed in your environment:

- MQSeriesRuntime
- MQSeriesServer

About this task

For information about installing Advanced Message Security follow the guidance for the appropriate platform.

Procedure

- "Installing Advanced Message Security on AIX" on page 521
- "Installing Advanced Message Security on HP-UX" on page 521
- "Installing Advanced Message Security on IBM i" on page 522
- "Installing Advanced Message Security on Linux" on page 522

- “Installing Advanced Message Security on Windows” on page 524

Installing Advanced Message Security on AIX:

You can install Advanced Message Security component on AIX platforms using either system management interface tool (SMIT) or the command line.

Installing using SMIT:

Procedure

1. Log on as root.
2. Change the directory to the location of the installation packages.
3. Start the system management interface tool (SMIT). The system management menu is displayed.
4. Select the required SMIT window using the following sequence:
   ```
   Software Installation and Maintenance
   Install and Update Software
   Install Software
   ```
5. Enter the directory location of the installation package.
6. Press F4 to list the software in the **SOFTWARE name** option.
7. Select the `mqm.ams.rte` and press Enter.
8. Accept the default setting for the remaining options and press Enter.

Results

Advanced Message Security has been installed successfully.

Installing using command line:

Procedure

1. Log on as root.
2. Set your current directory to the location of the installation file. The location might be the mount point of the DVD, a network location, or a local file system directory.
3. Run the following command:
   ```
   installp -a -c -Y -d. mqm.ams.rte
   ```
   Note the period, signifying the current directory, following the **`-d`** parameter.

Results

Advanced Message Security component has been installed successfully.

Installing Advanced Message Security on HP-UX:

You can install Advanced Message Security component on HP-UX platforms.

Procedure

1. Log on as root.
2. Set your current directory to the location of the installation file. The location might be the mount point of the DVD, a network location, or a local file system directory.
3. In the command line, issue the following command:
   ```
   swinstall -s MQSERIES.MQM-AMS
   ```

Results

Advanced Message Security component has been installed successfully.

Installing Advanced Message Security on IBM i: IBM i

You can install Advanced Message Security component on IBM i.

Procedure

Install AMS using the command:

```
RSTLICPGM LICPGM(5724H72) DEV(installation device) OPTION(2) OUTPUT(*PRINT)
```

where the parameters of **RSTLICPGM** are:

LICPGM(5724H72)
: The product identifier for IBM MQ for IBM i.

DEV(installation device)
: The device from which the product is to be loaded, typically an optical drive, for example, OPT01.

OPTION(2)
: Install Advanced Message Security for IBM i

OUTPUT(*PRINT)
: The output is printed with the spooled output of the job.

Results

The AMS component has been installed successfully.

Once AMS is installed on an IBM MQ server installation, any:
- Queue managers that are subsequently started enable security policy management features.
- Applications that connect to the queue manager enable interceptors.

What to do next

See Setting up certificates and the keystore configuration file on IBM i for details on setting up your security policy.

Installing Advanced Message Security on Linux: Linux

You can install Advanced Message Security on Linux platforms.

Procedure
1. Log on as root.
2. Set your current directory to the location of the installation file. The location might be the mount point of the server CD, a network share, or a local file system directory.
3. If this installation is not the first installation on the system, you must run the **crtmqpkg** command to create a unique set of packages to install on the system. In order for the **crtmqpkg** command to run on Linux, the **pax** command or **rpmbuild** must be installed.

   Important: **pax** and **rpmbuild** are not supplied as part of the product. You must obtain these from your Linux distribution supplier.
   a. Enter the following command:
   ```
   ./crtmqpkg suffix
   ```

where *suffix* is a name of your choosing, that uniquely identifies the installation packages on the system. *suffix* is not the same as an installation name, although the names can be identical. *suffix* is limited to 16 characters in the ranges A-Z, a-z, and 0-9.

Note: This command creates a full copy of the installation packages in a subdirectory of `/var/tmp`. You must ensure that the system has enough space before running the command.

b. Set your current directory to the location specified when the **`crtmqpkg`** command completes. This directory is a subdirectory of `/var/tmp/mq_rpms`, in which the unique set of packages is created. The packages have the *suffix* value contained within the filename. For example, using a suffix of "1":

```
./crtmqpkg 1
```

there is a subdirectory named `/var/tmp/mq_rpms/1/i386` and the packages are renamed, for example:

```
From: MQSeriesAMS-V.R.M-F.i386.rpm
To: MQSeriesAMS_1-V.R.M-F.i386.rpm
```

where:

V	Represents the version of the product that you are installing
R	Represents the release of the product that you are installing
M	Represents the modification of the product that you are installing
F	Represents the fix pack level of the product that you are installing

4. In the command line, issue the following command:

   This example shows a minimum installation:

   ```
   rpm -iv package_name
   ```

   where *`package_name`* is one of the following:

   - MQSeriesAMS-V.R.M-F.i386.rpm
   - MQSeriesAMS-V.R.M-F.x86_64.rpm
   - MQSeriesAMS-V.R.M-F.ppc.rpm
   - MQSeriesAMS-V.R.M-F.s390.rpm

Results

Advanced Message Security has been installed successfully.

Installing Advanced Message Security on Solaris: Solaris

You can install Advanced Message Security component on Solaris platforms.

Procedure

1. Log on as root.
2. If this installation is not the first installation on the system, you must run **`crtmqpkg`** to create a unique set of packages to install on the system:

   a. Enter the following command:

   ```
   ./crtmqpkg suffix
   ```

   where *suffix* is a name of your choosing, that will uniquely identify the installation packages on the system. *suffix* is not the same as an installation name, although the names can be identical. *suffix* is limited to 16 characters in the ranges A-Z, a-z, and 0-9.

   b. Set your current directory to the location specified when the **crtmqpkg** command completes. This directory is a sub-directory of `/var/spool`, in which the unique set of packages is created. The packages have the *suffix* value contained within the filename.
3. Start the installation process:
   - If the installation is the first installation on the system, enter the following command to start the installation process:
     ```
     pkgadd -d.
     ```
   - If the installation is not the first installation on the system, enter the following command to start the installation process:
     ```
     pkgadd mqm- suffix
     ```

     where *suffix* is the suffix chosen in the previous step.
4. You are prompted to choose a location for installation.
   - To install to the default location, `/opt/mqm`, enter `y`.
   - To install to a non-default directory, enter `n`. Then enter the required installation path, and confirm your choice.
5. Choose the `mqams` component.
6. If the path chosen in step 4 does not exist, you are asked if you want to create it. You must enter `y` to proceed.
7. A message is issued when the installation is complete. Enter `q` to exit the `pkgadd` program.

Results

Advanced Message Security component has been installed successfully.

Installing Advanced Message Security on Windows: Windows

You can install the Advanced Message Security component on Windows platforms.

Using the Launchpad:
Procedure
1. Access the IBM MQ installation image. The location might be the mount point of the DVD, a network location, or a local file system directory.
2. Locate `setup.exe` in the base directory of the IBM MQ installation image.
   - From a DVD, this location might be:
     ```
     E:\ setup.exe
     ```
   - From a network location, this location might be:
     ```
     m:\instmqs\ setup.exe
     ```
   - From a local file system directory, this location might be:
     ```
     C:\instmqs\ setup.exe
     ```
3. Double-click the **Setup** icon to start the installation process. It is possible to start the process by either:
   - Running `setup.exe` from the command prompt.
   - Double-clicking `setup.exe` from IBM MQ Explorer.

   Note: If you are installing on a Windows system with UAC enabled, accept the Windows prompt to allow the launchpad to run as elevated. During installation, you might also see Open File - Security Warning dialog boxes that list International Business Machines Limited as the publisher. Click **Run** to allow the installation to continue.

   The IBM MQ Installation Launchpad window is displayed.
4. Continue to follow the Launchpad instructions as shown on screen.

Uninstalling AMS on Multiplatforms

Multi

Use the information for your platform to uninstall the Advanced Message Security (AMS) component.

Procedure

- "Uninstalling AMS on AIX"
- "Uninstalling AMS on HP-UX" on page 526
- "Uninstalling AMS on Linux" on page 527
- "Uninstalling AMS on Windows" on page 528

Related tasks:

"Installing AMS on Multiplatforms" on page 520
Use the information for your platform to guide you through installing the Advanced Message Security (AMS) component.

Uninstalling AMS on AIX: AIX

On AIX platforms, you can remove Advanced Message Security component either using SMIT or the command line.

Procedure

1. Stop all IBM MQ applications associated with the installation you are uninstalling.
2. For a server installation, end any IBM MQ activity associated with the installation you are uninstalling:
   a. Log in as a user in the group mqm.
   b. Set up your environment to work with the installation you want to uninstall. Enter the following command:
   ```
   . MQ_INSTALLATION_PATH/bin/setmqenv
   ```
   where . MQ_INSTALLATION_PATH refers to the location where IBM MQ is installed.
   c. Display the state of all queue managers on the system. Enter the following command:
   ```
   dspmq -o installation
   ```
   d. Stop all running queue managers associated with the installation you want to uninstall. Enter the following command for each queue manager:
   ```
   endmqm QMgrName
   ```
   e. Stop any listeners associated with the queue managers. Enter the following command for each queue manager:
   ```
   endmqlsr -m QMgrName
   ```
3. Log in as root.
4. Uninstall AMS component using either **installp** or **smit**. If AMS component was installed in a non-default location, you must use **installp** to uninstall.
   - Uninstall using **installp** by entering one of the following commands:
     - For an installation in the default location /usr/mqm
     ```
     installp -u mqm.ams.rte
     ```
     - For an installation in a non-default location:
     ```
     installp -R
     usil -u mqm.ams.rte
     ```
     where *usil* is the path of the User Specified Installation Location (USIL) specified when the product was installed.

- Uninstall using **smit**:
  a. Select the required **smit** window using the following sequence:
  ```
  Software Installation and Maintenance
  Software Maintenance and Utilities
  Remove Installed Software
  ```
  b. List the software in the **SOFTWARE name** field:
     1) Enter **.**
     2) Press **F4**
  c. Select the file sets to uninstall from the list (those beginning with mqm), and press **Enter**. There is an option at this stage to do a preview. Leave the option set to the default value of **Yes** to preview the file sets you are uninstalling, or select **No** to not preview these file sets.
  d. Press **Enter** on the **Remove Installed Software** panel, it asks whether you are sure, press **Enter**.

Results

The Advanced Message Security component has been uninstalled.

Uninstalling AMS on HP-UX: HP-UX

Use the `swremove` command to remove Advanced Message Security component on HP-UX platforms.

Procedure

1. Stop all IBM MQ applications associated with the installation you are uninstalling.
2. For a server installation, end any IBM MQ activity associated with the installation you are uninstalling:
   a. Log in as a user in the group `mqm`.
   b. Set up your environment to work with the installation you want to uninstall. Enter the following command:
   ```
   . MQ_INSTALLATION_PATH/bin/setmqenv
   ```
   where `. MQ_INSTALLATION_PATH` refers to the location where IBM MQ is installed.
   c. Display the state of all queue managers on the system. Enter the following command:
   ```
   dspmq -o installation
   ```
   d. Stop all running queue managers associated with the installation you want to uninstall. Enter the following command for each queue manager:
   ```
   endmqm QMgrName
   ```
   e. Stop any listeners associated with the queue managers. Enter the following command for each queue manager:
   ```
   endmqlsr -m QMgrName
   ```
3. Log on as root.
4. Run the following command:
   ```
   swremove MQSERIES.MQM-AMS
   ```

Results

The Advanced Message Security component has been uninstalled.

Uninstalling AMS on Linux: Linux

Use the `rpm` command to remove Advanced Message Security component on Linux platforms.

Procedure

1. Stop all IBM MQ applications associated with the installation you are uninstalling.
2. For a server installation, end any IBM MQ activity associated with the installation you are uninstalling:
   a. Log in as a user in the group `mqm`.
   b. Set up your environment to work with the installation you want to uninstall. Enter the following command:
      ```
      . MQ_INSTALLATION_PATH/bin/setmqenv
      ```
      where `. MQ_INSTALLATION_PATH` refers to the location where IBM MQ is installed.
   c. Display the state of all queue managers on the system. Enter the following command:
      ```
      dspmq -o installation
      ```
   d. Stop all running queue managers associated with the installation you want to uninstall. Enter the following command for each queue manager:
      ```
      endmqm QMgrName
      ```
   e. Stop any listeners associated with the queue managers. Enter the following command for each queue manager:
      ```
      endmqlsr -m QMgrName
      ```
3. Log in as root.
4. Run the following command:
   ```
   rpm -e package_name
   ```
   where *package_name* is MQSeriesAMS-*V.R.M-F*

V	Represents the version of the product that you are uninstalling
R	Represents the release of the product that you are uninstalling
M	Represents the modification of the product that you are uninstalling
F	Represents the fix pack level of the product that you are uninstalling

Results

The Advanced Message Security component has been uninstalled.

Uninstalling on Solaris: Solaris

Use the pkgrm to uninstall IBM MQ on Solaris.

About this task

Restriction: On Solaris, you cannot remove components from an installation. There is no supported method of doing this.

For information about uninstalling IBM MQ on Solaris, see “Uninstalling IBM MQ on Solaris” on page 434.

Uninstalling AMS on Windows: Windows

You can uninstall Advanced Message Security component using the GUI uninstallation wizard, or a command-line interface.

Using the installation wizard:
Procedure

1. Insert the IBM MQ Server DVD into the DVD-ROM drive.
2. If autorun is enabled, the installation process starts.

   Otherwise, double-click the **Setup** icon in the root folder of the DVD to start the installation process.

   The IBM MQ Installation Launchpad window is displayed.
3. Click the **IBM MQ Installation**.
4. Click **Launch IBM MQ Installer**. Click **Next** until the IBM MQ Program Maintenance panel is displayed with a welcome message.

   If this panel is not displayed, IBM MQ for Windows, Version 7.5 is not installed on this machine. When presented with the option, select to remove/maintain or upgrade.
5. Select **Maintain or upgrade an existing instance**, then click **Next**.
6. If there are any existing queue managers, the Removing Server feature panel is displayed.

   Click one of the following options, then click **Next**:
   - **Keep** - keep existing queue managers and their objects.
   - **Remove** - remove existing queue managers and their objects.

   The Program Maintenance panel is displayed, with a summary of the installation to be removed.
7. Click **Modify** and click **Next**.
8. On the list of available IBM MQ features, click Advanced Message Security, select **Do not install this feature (remove if already intalled)**, and click **Next**. The Ready to modify IBM MQ panel appears with the summary of your changes.
9. Click **Modify** and **Next** on the following panel to continue.

Results

Selected features of Advanced Message Security component have been removed.

Installing Managed File Transfer

MQ Adv.

From Version 7.5 onwards, Managed File Transfer is installed as a component of IBM MQ on UNIX, Linux, and Windows. Managed File Transfer remains as a separate product on IBM i and z/OS.

Before you begin

Before you install Managed File Transfer, check that your system meets both the hardware and software requirements of the product. See System Requirements for IBM MQ.

For all platforms, you must have one IBM WebSphere MQ Version 7.0, or higher, queue manager available in your Managed File Transfer network to use as the coordination queue manager.

.

About this task

ULW The following steps describe installing Managed File Transfer as a component of IBM MQ on UNIX, Linux, and Windows.

Procedure

1. Decide which Managed File Transfer components to install. Managed File Transfer can be installed as four different options, depending on your operating system and overall setup. These options are Managed File Transfer Agent, Managed File Transfer Service, Managed File Transfer Logger, or Managed File Transfer Tools.

   To decide which components to install, review the product options and topology information in the following topics:
   - Managed File Transfer product options
   - Managed File Transfer topology overview
2. Install IBM MQ, including Managed File Transfer components. For information about which specific components to install for your platform, including Managed File Transfer, see "IBM MQ components and features" on page 192.

   For more information about installing IBM MQ on UNIX, Linux, and Windows, see the appropriate information for your platform:
   - AIX "Installing and uninstalling IBM MQ on AIX" on page 215
   - HP-UX "Installing and uninstalling IBM MQ on HP-UX" on page 263
   - Linux "Installing and uninstalling IBM MQ on Linux" on page 330
   - Solaris "Installing and uninstalling IBM MQ on Solaris" on page 398
   - Windows "Installing and uninstalling IBM MQ on Windows" on page 435

Related information:

Managed File Transfer

IBM i Installing Managed File Transfer on IBM i
Install IBM MQ Managed File Transfer for IBM i by installing IBM MQ Java Messaging and Web Services server in its primary language, and installing additional options.

z/OS Installing IBM MQ for z/OS
You install Managed File Transfer on your IBM MQ for z/OS system by using SMP/E.

Managed File Transfer topology overview

Installed command sets

Managed File Transfer product options

ULW

Managed File Transfer can be installed as four different options, depending on your operating system and overall setup. These options are Managed File Transfer Agent, Managed File Transfer Service, Managed File Transfer Logger, or Managed File Transfer Tools.

Managed File Transfer Agent

A file transfer agent connects to an IBM MQ queue manager, and transfers file data, as messages, to other file transfer agents.

You install an agent through either the Managed File Transfer Agent or Managed File Transfer Service installation options.

The Managed File Transfer Agent option installs an agent that has the following capabilities:

- Make client or bindings mode connections to queue managers.

  Note: When the file transfer agent and queue manager are on the same system, consider using the bindings mode connections.
- Transfer files to and from other Managed File Transfer agents.
- Transfer files to and from Connect:Direct® nodes.

The Managed File Transfer Service option, described in the next section, installs a file transfer agent that also has additional capability to transfer files to and from legacy FTP, FTPS, or SFTP protocol servers.

Managed File Transfer Service

The Managed File Transfer Service option installs an agent that has the following capabilities:

- Make client or bindings mode connections to queue managers.

  Note: When the file transfer agent and queue manager are on the same system, consider using the bindings mode connections.
- Transfer files to and from other Managed File Transfer agents.
- Transfer files to and from Connect:Direct nodes.
- Create protocol bridge agents that transfer files to and from legacy SFTP, FTP, or FTPS protocol servers.

Some capabilities are available on only a subset of supported platforms. For more information, see IBM MQ System Requirements.

A Managed File Transfer Service can only be installed on systems where the IBM MQ Server option is already installed.

Managed File Transfer Logger

A file transfer logger connects to an MQ queue manager, often the queue manager that is designated as the coordination queue manager, and logs audit-related file transfer data to either a database or a file. A logger can only be installed on systems where the IBM MQ Server installation option is already installed.

Managed File Transfer Tools

The Managed File Transfer Tools are command line tools that you use to interact with file transfer agents. The tools allow you to start file transfers, schedule file transfers and create resource monitors from the command line. The Managed File Transfer Tools need not be installed on the same system as the file transfer agents that they interact with.

UNIX

Managed File Transfer Base

On UNIX platforms, there is an additional Managed File Transfer Base installation component. This component contains files common to all of the installation options. You must install the Managed File Transfer Base component before installing any of the Agent, Logger, Service, or Tools components.

For more information about the IBM MQ components that are required for each product option on UNIX platforms, see the following topics:

- AIX “Required MFT components on AIX”
- HP-UX “Required MFT components on HP-UX” on page 532
- Linux “Required MFT components on Linux” on page 533
- Solaris “Required MFT components on Solaris” on page 534

Related information:

Managed File Transfer introduction

Managed File Transfer topology overview

Required MFT components on AIX: AIX

Managed File Transfer can be installed as four different options, depending on your operating system and overall setup. On AIX systems, these options are Managed File Transfer Agent, Managed File Transfer Logger, Managed File Transfer Service, and Managed File Transfer Tools, and each option requires specific components.

Managed File Transfer Agent

mqm.base.runtime

mqm.java.rte

mqm.jre.rte

mqm.ft.base

mqm.ft.agent

Managed File Transfer Logger

mqm.base.runtime

mqm.server.rte

mqm.java.rte

mqm.jre.rte

mqm.ft.base

mqm.ft.logger

Managed File Transfer Service

mqm.base.runtime

mqm.server.rte

mqm.java.rte

mqm.jre.rte

mqm.ft.base

mqm.ft.agent

mqm.ft.service

Managed File Transfer Tools

mqm.base.runtime

mqm.java.rte

mqm.jre.rte

mqm.ft.base

mqm.ft.tools

Required MFT components on HP-UX: HP-UX

Managed File Transfer can be installed as four different options, depending on your operating system and overall setup. On HP-UX systems, these options are Managed File Transfer Agent, Managed File Transfer Logger, Managed File Transfer Service, and Managed File Transfer Tools, and each option requires specific components.

Managed File Transfer Agent

MQSERIES.MQM-RUNTIME

MQSERIES.MQM-JAVA

MQSERIES.MQM-JAVAJRE

MQSERIES.MQM-FTBASE

MQSERIES.MQM-FTAGENT

Managed File Transfer Logger

MQSERIES.MQM-RUNTIME

MQSERIES.MQM-SERVER

MQSERIES.MQM-JAVA

MQSERIES.MQM-JAVAJRE

MQSERIES.MQM-FTBASE

MQSERIES.MQM-FTLOGGER

Managed File Transfer Service

MQSERIES.MQM-RUNTIME

MQSERIES.MQM-SERVER

MQSERIES.MQM-JAVA

MQSERIES.MQM-JAVAJRE

MQSERIES.MQM-FTBASE

MQSERIES.MQM-FTAGENT

MQSERIES.MQM-FTSERVICE

Managed File Transfer Tools

MQSERIES.MQM-RUNTIME

MQSERIES.MQM-JAVA

MQSERIES.MQM-JAVAJRE

MQSERIES.MQM-FTBASE

MQSERIES.MQM-FTTOOLS

Required MFT components on Linux: Linux

Managed File Transfer can be installed as four different options, depending on your operating system and overall setup. On Linux systems, these options are Managed File Transfer Agent, Managed File Transfer Logger, Managed File Transfer Service, and Managed File Transfer Tools, and each option requires specific components.

Managed File Transfer Agent

MQSeriesRuntime

MQSeriesJava

MQSeriesJRE

MQSeriesFTBase

MQSeriesFTAgent

Managed File Transfer Logger

MQSeriesRuntime

MQSeriesServer

MQSeriesJava

MQSeriesJRE

MQSeriesFTBase

MQSeriesFTLogger

Managed File Transfer Service

MQSeriesRuntime

MQSeriesServer

MQSeriesJava

MQSeriesJRE

MQSeriesFTBase

MQSeriesFTAgent

MQSeriesFTService

Managed File Transfer Tools

MQSeriesRuntime

MQSeriesJava

MQSeriesJRE

MQSeriesFTBase

MQSeriesFTTools

Required MFT components on Solaris: Solaris

Managed File Transfer can be installed as four different options, depending on your operating system and overall setup. On Solaris systems, these options are Managed File Transfer Agent, Managed File Transfer Logger, Managed File Transfer Service, and Managed File Transfer Tools, and each require specific components.

Managed File Transfer Agent

runtime

java

jre

ftbase

ftagent

Managed File Transfer Logger

runtime

server

java

jre

ftbase

ftlogger

Managed File Transfer Service

runtime

server

java

jre

ftbase

ftagent

ftservice

Managed File Transfer Tools

runtime

java

jre

ftbase

fttools

Installed MFT command sets

The following table shows which Managed File Transfer commands are installed with each component.

Table 74. Managed File Transfer commands available in each command set

Command	Agent command set	Service command set	Tools command set	Logger command set	V 9.0.1 Redistributable Managed File Transfer Agent set
fteAnt			✓		✓
fteBundleConfiguration			✓ (UNIX, Linux, and Windows only)		✓
fteCancelTransfer			✓		✓
fteChangeDefaultConfigurationOpti	✓	✓	✓	✓	✓
fteCleanAgent	✓	✓			✓
fteCreateAgent	✓	✓			✓
fteCreateBridgeAgent		✓			✓
fteCreateCDAgent	✓ (UNIX, Linux, and Windows only)	✓ (UNIX, Linux, and Windows only)			
V 9.0.1 fteCreateEnvironment					✓
fteCreateLogger				✓	
fteCreateMonitor			✓		✓
fteCreateTemplate			✓		✓
fteCreateTransfer			✓		✓
fteDefine			✓ (UNIX, Linux, and Windows only)		✓
fteDelete			✓ (UNIX, Linux, and Windows only)		✓
fteDeleteAgent	✓	✓			✓
fteDeleteLogger				✓	
fteDeleteMonitor			✓		✓
fteDeleteScheduledTransfer			✓		✓

Table 74. Managed File Transfer commands available in each command set *(continued)*

Command	Agent command set	Service command set	Tools command set	Logger command set	V 9.0.1 Redistributable Managed File Transfer Agent set
fteDeleteTemplates			✓		✓
fteDisplayVersion	✓	✓		✓	✓
fteListAgents	✓	✓	✓	✓	✓
fteListMonitors			✓		✓
fteListScheduledTransfers			✓		✓
fteListTemplates			✓		✓
fteMigrateAgent	✓	✓			✓
fteMigrateConfigurationOptions	✓	✓	✓	✓	✓
fteMigrateLogger				✓	
fteModifyAgent	✓ (Windows only)	✓ (Windows only)			✓
fteModifyLogger				✓ (Windows only)	
fteObfuscate	✓	✓		✓	✓
ftePingAgent			✓		✓
fteRAS		✓			✓
fteSetAgentLogLevel	✓				✓
fteSetAgentTraceLevel	✓	✓			✓
fteSetLoggerTraceLevel				✓	
fteSetupCommands	✓	✓	✓	✓	✓
fteSetupCoordination	✓	✓	✓	✓	✓
fteShowAgentDetails	✓	✓	✓	✓	✓
fteShowLoggerDetails				✓	
fteStartAgent	✓	✓			✓
fteStartLogger				✓	

Table 74. Managed File Transfer commands available in each command set (continued)

Command	Agent command set	Service command set	Tools command set	Logger command set	V 9.0.1 Redistributable Managed File Transfer Agent set
fteStopAgent	✓	✓			✓
fteStopLogger				✓	

Installing MQ Telemetry

AIX Linux Windows MQ Adv.

Installation tasks associated with MQ Telemetry are grouped in this section.

About this task

MQ Telemetry is installed as part of the IBM MQ server installation.

MQ Telemetry is a separately installed component of IBM MQ and is another option on the IBM MQ installer. Make sure that you purchase a license for using IBM MQ Advanced before the installation (see IBM MQ license information).

Procedure

Install IBM MQ, including MQ Telemetry. For information about which specific components to install for your platform, including MQ Telemetry, see “IBM MQ components and features” on page 192.
For more information about installing IBM MQ on AIX, Linux, or Windows, see the appropriate information for your platform:

- AIX “Installing and uninstalling IBM MQ on AIX” on page 215
- Linux “Installing and uninstalling IBM MQ on Linux” on page 330
- Windows “Installing and uninstalling IBM MQ on Windows” on page 435

Installation considerations for MQ Telemetry

From IBM WebSphere MQ Version 7.1, MQ Telemetry is a component of the main IBM MQ product, and is no longer a separate plugin. You can choose to install MQ Telemetry when you first install IBM MQ, or when you modify an existing IBM MQ installation. To get a set of client libraries that help you write messaging applications for telemetry, download the IBM Messaging Telemetry Clients SupportPac.

MQ Telemetry overview

See Introduction to MQ Telemetry for general details about MQ Telemetry.

Support for IBM MQ Explorer

You can use IBM MQ Explorer to configure and manage the MQ Telemetry runtime component. For a queue manager to accept connections from a telemetry device, one or more telemetry channels are needed. To enable MQTT, there is a `define sample configuration` wizard that can be run from IBM MQ Explorer. The wizard runs through a series of steps including defining and starting the telemetry (MQXR) service, setting up the default transmission queue, and configuring a telemetry channel. For more information about using the `define sample configuration` wizard, and any implications, see “Verifying the installation of MQ Telemetry by using IBM MQ Explorer” on page 540.

The IBM MQ Explorer support provides the following capabilities:

- Telemetry node and content panel - providing welcome information, define sample configuration wizard, run MQTT client utility, Help on MQ Telemetry, and status information about the MQ Telemetry Service.
- Define sample configuration wizard - quickly configures a queue manager to support MQTT.
- New Telemetry Channel wizard - gathers information required to create a telemetry channel object.
- Telemetry Channels node and content panel - displays telemetry channels in the IBM MQ Explorer Content view.
- Telemetry Channel Status node and content panel - displays telemetry channel status in the IBM MQ Explorer Content view.
- MQTT Client Utility - provides a simple GUI for publishing and subscribing to topics.
- Help on MQ Telemetry.

You can install the MQ Telemetry runtime component on one system and configure and manage it using the IBM MQ Explorer installed on another system. However, the components can be installed only on systems with the appropriate prerequisites. For information about these prerequisites, see IBM MQ system requirements.

MQ Telemetry client libraries and SDK

To help you write messaging applications for MQTT networks, you can install and use a set of free client libraries. After you develop your applications, these applications and the client libraries are then deployed together to the appropriate system.

In previous versions of IBM MQ, the client libraries were supplied with the product, in the Client Software Development Kit (SDK). From IBM MQ Version 8.0, this SDK is no longer supplied as part of the product. Instead, the current version of the SDK is available as the IBM Messaging Telemetry Clients SupportPac.

To use the telemetry MQTT clients, download the IBM Messaging Telemetry Clients SupportPac, then install the clients in a directory of your own choosing. The sample applications and client libraries are in client-specific directories under `CLIENTPACKDIR/SDK/clients`, where *CLIENTPACKDIR* is the directory in which you decompressed the client pack. Note that the example scripts included in IBM MQ assume that the clients are in the directories specified in Location of telemetry logs, error logs, and configuration files.

The IBM Messaging Telemetry Clients SupportPac provides you with the following resources:

- Sample MQTT client applications written in Java, in JavaScript, and in C.
- MQTT client libraries that support these client applications, and enable them to run on most platforms and devices, including Android devices and products from Apple.

Related information:
MQ Telemetry
Telemetry use cases
Administering MQ Telemetry
Developing applications for MQ Telemetry
MQ Telemetry reference
MQ Telemetry troubleshooting

Verifying the installation of MQ Telemetry

There are three ways to verify the installation of MQ Telemetry. Any can be used, regardless of whether MQ Telemetry was installed as a custom installation of IBM MQ, or added to an existing installation of IBM MQ.

About this task

Within IBM MQ you can verify the installation of MQ Telemetry either by using IBM MQ Explorer, or by using the command line.

You can also verify the installation by using the MQTT messaging client for JavaScript in a browser that supports the RFC 6455 (WebSocket) standard. A version of this client is installed with MQ Telemetry, and the latest version is available as part of the IBM Messaging Telemetry Clients SupportPac. To verify the MQ Telemetry installation you do not need the latest version of the client, so you should omit step 1.

Procedure

Verify your installation in one of the following ways:
- By using IBM MQ Explorer as described in "Verifying the installation of MQ Telemetry by using IBM MQ Explorer."
- By using the command line as described in "Verifying the installation of MQ Telemetry using the command line" on page 542.

Verifying the installation of MQ Telemetry by using IBM MQ Explorer: Linux Windows

Use the Define sample configuration wizard and the MQTT client utility in IBM MQ Explorer to verify that the MQ Telemetry components have installed. Also check that publish/subscribe works correctly.

Before you begin

The MQ Telemetry runtime and support for IBM MQ Explorer must be installed. The telemetry folder is part of a queue manager. To view the telemetry folder, you must start a queue manager.

Before running the define sample configuration wizard on an existing queue manager, review the information provided by the wizard about the configuration changes that are made. The changes might have implications for the configuration of the existing queue manager. Alternatively, run the sample configuration wizard on a newly created queue manager to avoid changing any security settings.

About this task

To configure MQ Telemetry there is a define sample configuration wizard that can be run from IBM MQ Explorer. The wizard runs through a series of steps, including defining and starting the telemetry (MQXR) service, setting up the default transmission queue, and configuring a telemetry channel.

If you would prefer to do this manually, see Configuring a queue manager for telemetry on Linux and AIX. For Windows, see Configuring a queue manager for telemetry on Windows.

You can open the define sample configuration wizard from the MQ Telemetry Welcome page in IBM MQ Explorer. The wizard determines which steps are required based on the current configuration.

For example, the following actions might be specified by the wizard:
- Define the telemetry (MQXR) service.
- Start the telemetry (MQXR) service.
- Define the telemetry transmit queue.
- Set the default transmit queue of the queue manager to `SYSTEM.MQTT.TRANSMIT.QUEUE`.

If telemetry is already configured for this queue manager, the link to open the wizard is replaced with static text. The text confirms that the sample configuration has been set up.

After the configuration has finished, you can use IBM MQ Explorer to open the MQTT client utility. Use the MQTT client utility to verify that MQ Telemetry is set up correctly.

The following items summarize the main goals that can be achieved using the MQTT client utility:
- Validation of a basic or custom MQ Telemetry configuration by connecting, subscribing to topics and publishing messages.
- Showcases the main features of MQTT protocol.
- Provides a simple tool to aid in debugging MQ Telemetry applications.

You can find additional information within the IBM MQ Explorer by using the **Help** menu or pressing the **F1** key.

Procedure

1. Start IBM MQ Explorer.

   On Windows and Linux systems, you can start IBM MQ Explorer by using the system menu, the `MQExplorer` executable file, the **`mqexplorer`** command, or the **`strmqcfg`** command.
2. Open the Welcome to MQ Telemetry page.
   - To use an existing queue manager, click on `IBM MQ\Queue Managers\`*`qMgrName`*`\Telemetry` folder to open the Welcome to MQ Telemetry page.
   - If, for the reasons mentioned, you decide to use a new queue manager,
     a. Click **Queue Managers** > **New** > **Queue Manager**.
     b. Type `MQTTVerification` as the **Queue manager name** > **Next** > **Next** > **Next**.
     c. Change the default port in **Listen on port number**, if the port is in use > **Finish**.
     d. When the queue manager starts, click on `IBM MQ\Queue Managers\MQTTVerification\Telemetry` folder to open the Welcome to MQ Telemetry page.
3. From the Welcome to MQ Telemetry page in IBM MQ Explorer, click **Define sample configuration**.

   If this link is not present, and instead you see the text, “The sample configuration has been set up for this queue manager”, then telemetry has already been configured. Proceed to step 6.

   If you clicked **Define sample configuration**, the page opens, and lists actions that are to be performed as part of the sample configuration.
4. Leave **Launch MQTT client utility** checked, if you want to automatically start the MQTT client utility. The check box is selected by default.
5. Click **Finish**.
6. Click **Connect**.

   In the MQTT client utility panel, ensure that the host and port names are correct.

   If you did not automatically start the MQTT client utility panel in step 4, you can start it either by using a direct link from the Welcome to MQ Telemetry panel, or by right-clicking a NON-TLS channel, which allows you to control the channel it runs on.

The client history records a `Connected` event.

7. Click **Subscribe**.

   The client history records a `Subscribed` event.

8. Click **Publish**.

   The client history records a `Published` and `Received` event.

Results

If the publish/subscribe finishes successfully, the MQ Telemetry installation is verified.

If you encounter problems during the installation process, view the error log:

- On Windows, the default location for this log is, *`IBM MQ data directory`*`\qmgrs\`*`qMgrName`*`\mqxr`
- On AIX and Linux, the default location for this log is, `/var/mqm/qmgrs/`*`qMgrName`*`/mqxr/`

Verifying the installation of MQ Telemetry using the command line:

Follow these instructions to run scripts and a sample application to verify that the MQ Telemetry components have installed, and are able to publish and subscribe.

Before you begin

The telemetry (MQXR) service must be started to run the sample programs. The user ID must be a member of the `mqm` group.

The `SampleMQM` script creates and uses a queue manager called `MQXR_SAMPLE_QM`. Therefore, do not run unaltered on a system that already has a `MQXR_SAMPLE_QM` queue manager. Any changes made might have implications for the configuration of the existing queue manager.

This task uses the MQTTV3 sample Java application, and the associated Java client library. These resources are no longer included in MQ Telemetry, they are now part of the IBM Messaging Telemetry Clients SupportPac. You install this SupportPac as part of this task.

There are two commands to run the MQTTV3 sample Java application. The first command creates a subscription, then waits for a message. The second command publishes to that subscription. Therefore the commands must be entered into different command lines or shell windows.

About this task

To perform verification on a server or device without a GUI, scripts are provided in the samples directory. The `SampleMQM` script performs the required steps to configure MQ Telemetry. The MQTTV3 sample Java application can then be run to validate the basic or custom MQ Telemetry configuration by connecting, subscribing to topics, and publishing messages. The `CleanupMQM` sample script can be run to delete the queue manager created by the `SampleMQM` script.

The following items summarize the main goals that can be achieved using this verification procedure:

- Validate a basic or custom MQ Telemetry configuration by connecting, subscribing to topics and publishing messages.
- Showcase the main features of the MQTT protocol.
- Provide a simple tool to aid in debugging MQ Telemetry applications.

Procedure

1. Download the IBM Messaging Telemetry Clients SupportPac.
2. Decompress the client pack into a directory of your own choosing.

The sample applications and client libraries are in client-specific directories under *CLIENTPACKDIR*/SDK/clients, where *CLIENTPACKDIR* is the directory in which you decompressed the client pack. This task uses the MQTTV3 sample Java application, and the associated Java client library. These resources are available in the following location:

```
CLIENTPACKDIR/SDK/clients/java
```

3. Configure MQ Telemetry.

   The SampleMQM script runs through a series of steps, including creating the MQXR_SAMPLE_QM queue manager, defining and starting the telemetry (MQXR) service, setting up the default transmission queue, and configuring a telemetry channel.

   For information about performing this manually, see Configuring a queue manager for telemetry on Linux and AIX , or Configuring a queue manager for telemetry on Windows .

   - On Windows systems, enter the following command in a command line:
     ```
     MQINSTDIR\mqxr\samples\SampleMQM.bat
     ```
   - On AIX or Linux systems, enter the following command in a shell window:
     ```
     MQINSTDIR/mqxr/samples/SampleMQM.sh
     ```

   where *MQINSTDIR* is the installation directory for this installation of IBM MQ. A queue manager called MQXR_SAMPLE_QM is created, and MQ Telemetry is configured.
4. Run the MQTTV3 sample Java application to create a subscription.
   - On Windows systems, enter the following commands in a command line:
     ```
     java -cp
     "CLIENTPACKDIR\SDK\clients\java\org.eclipse.paho.sample.mqttv3app.jar;
     CLIENTPACKDIR\SDK\clients\java\org.eclipse.paho.client.mqttv3.jar"
     org.eclipse.paho.sample.mqttv3app.Sample -a subscribe
     ```
   - On AIX or Linux systems, enter the following commands in a shell window:
     ```
     java -cp
     CLIENTPACKDIR/SDK/clients/java/org.eclipse.paho.sample.mqttv3app.jar:
     CLIENTPACKDIR/SDK/clients/java/org.eclipse.paho.client.mqttv3.jar
     org.eclipse.paho.sample.mqttv3app.Sample -a subscribe
     ```

   The subscription is created, and waits to receive a message.
5. Run the MQTTV3 sample Java application to publish to the subscription.
   - On Windows systems, enter the following command in a second command line:
     ```
     java -cp
     "CLIENTPACKDIR\SDK\clients\java\org.eclipse.paho.sample.mqttv3app.jar;
     CLIENTPACKDIR\SDK\clients\java\org.eclipse.paho.client.mqttv3.jar"
     org.eclipse.paho.sample.mqttv3app.Sample -m "Hello from an MQTT v3 application"
     ```
   - On AIX or Linux systems, enter the following command in a second shell window:
     ```
     java -cp
     CLIENTPACKDIR/SDK/clients/java/org.eclipse.paho.sample.mqttv3app.jar:
     CLIENTPACKDIR/SDK/clients/java/org.eclipse.paho.client.mqttv3.jar
     org.eclipse.paho.sample.mqttv3app.Sample -m "Hello from an MQTT v3 application"
     ```

   The message Hello from an MQTT v3 application, that you typed into the second command line or shell window, is published by that application and received by the application in the first window. The application in the first window shows it on the screen.
6. Press **Enter** in the first command line or shell window to end the subscribing application.
7. Remove the queue manager created by the SampleMQM script.
   - On Windows systems, enter the following command in a command line:
     ```
     MQINSTDIR\mqxr\samples\CleanupMQM.bat
     ```
   - On AIX or Linux systems, enter the following command in a shell window:
     ```
     MQINSTDIR/mqxr/samples/CleanupMQM.sh
     ```

Results

If the scripts finished, and messages can be sent and received, the MQ Telemetry installation is verified.

What to do next

If you encounter any problems during the verification process, see MQ Telemetry troubleshooting. You can also view the error log:

- On Windows systems, the default location for the queue manager log is `*MQINSTDIR*\qmgrs\MQXR_SAMPLE_QM\mqxr`
- On AIX and Linux systems, the default location for the queue manager log is `/var/mqm/qmgrs/MQXR_SAMPLE_QM/mqxr/`

Uninstalling MQ Telemetry components

AIX Linux Windows

Remove MQ Telemetry components from your computer.

Before you begin

Check that your user ID has the following authority to complete uninstallation tasks:

- Windows On all Windows operating systems and editions, your user ID must be a member of the Administrators group.
- AIX Linux On AIX and Linux systems, your user ID must have root authority to complete installation. Follow your local security guidelines to acquire root authority; either login as root, or log in as another user and become root.

Procedure

- Windows On Windows, to uninstall MQ Telemetry you must modify the existing IBM MQ. Deselect MQ Telemetry from the list of components that is installed in the main installation.
- AIX On AIX, run the command **`installp -u mqm.xr.service`**. For more information, refer to “Uninstalling or modifying IBM MQ on AIX” on page 247, but replace step 4 with the command in this step.
- Linux On Linux, run the command **`rpm -e MQSeriesXRService`**. For more information refer to “Uninstalling or modifying IBM MQ on Linux using rpm” on page 395, but replace step 4 with the command in this step.

Installing RDQM (replicated data queue managers)

Linux V 9.0.4 MQ Adv.

Installation tasks associated with RDQM are grouped in this section. RDQM is only available on RHEL 7.*x* on x86-64.

Before you begin

Pacemaker is one of the prerequisites for RDQM that you install as part of this task. Pacemaker requires that the following Linux packages are installed on the system:

- OpenIPMI-modalias.x86_64
- OpenIPMI-libs.x86_64
- libyaml.x86_64
- PyYAML.x86_64
- libesmtp.x86_64
- net-snmp-libs.x86_64
- net-snmp-agent-libs.x86_64
- openhpi-libs.x86_64
- libtool-ltdl.x86_64
- perl-TimeDate.x86_64

About this task

To install support for RDQM (replicated data queue managers), you perform the following tasks:

1. Install IBM MQ on each node.
2. Install DRBD and Pacemaker on each node.
3. Install RDQM on each node.
4. Configure the firewall on each node.

The DRBD and Pacemaker RPM packages are supplied on the IBM MQ media. You should install the versions supplied with IBM MQ. Do not download your own versions.

The DRBD and Pacemaker packages are signed with the LINBIT GPG key. Use the following command to import the public LINBIT GPG key:

```
rpm --import https://packages.linbit.com/package-signing-pubkey.asc
```

Without this step, an RPM install of these packages issues the following warnings:

```
warning: rpm-name: Header V4 DSA/SHA1 Signature, key ID 282b6e23: NOKEY"
```

The same installation location should be used on all three servers that are the nodes in the HA group, or both servers in a DR pair. You can have multiple IBM MQ installations on each server, but only one of these installations should be an RDQM installation.

The following installation script is supplied, you must run the script as `root`:

installRDQMsupport

By default, installs the IBM MQ runtime, server, samples, client, and RDQM RPM packages. Also installs the DRBD and Pacemaker RPM packages. (You can edit this file, if required, to add additional RPM packages to install; see "IBM MQ rpm components for Linux systems" on page 341 for a list of available RPM packages.)

A script for configuring the firewall for HA RDQM is supplied in the IBM MQ samples directory, you must run the script as `root`:

`MQ_INSTALLATION_PATH``/samp/rdqm/firewalld/configure.sh`

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)

V 9.0.5 For DR RDQM you specify the replication port when you create a DR RDQM, so must configure your firewall to add firewallD service rules accordingly.

Procedure

To install RDQM support, on each node:

1. Run the script `installRDQMsupport` to install IBM MQ, RDQM support, DRBD, and Pacemaker.
2. Run the script *`MQ_INSTALLATION_PATH`*`/samp/rdqm/firewalld/configure.sh` to configure the firewall for HA RDQM operation.

What to do next

You can now configure the Pacemaker cluster and replicated data queue managers, see RDQM high availability. Or you can configure disaster recovery replicated data queue managers, see RDQM disaster recovery.

Related information:

Migrating replicated data queue managers

Uninstalling RDQM (replicated data queue managers)

Linux V 9.0.4 MQ Adv.

You can uninstall RDQM by using the supplied uninstallation scripts.

About this task

Before you uninstall HA RDQM, you must first suspend the HA group, or remove it from the node altogether.

An uninstallation script is supplied in the root directory of the installation image, you must run the script as `root`:

`uninstallRDQMsupport`

By default, uninstalls the MQSeries Runtime, Server, Samples, Client and RDQM RPM packages along with the DRBD and Pacemaker RPM packages. You can edit the script, if required.

A script for undoing the firewall configuration is supplied in the IBM MQ samples directory, you must run the script as `root`:

`MQ_INSTALLATION_PATH``/samp/rdqm/firewalld/unconfigure.sh`

Removes firewallD service rules for DRBD, Pacemaker and IBM MQ.

Procedure

- To uninstall HA RDQM support, on each node:
  1. Suspend or delete the HA group. To suspend the HA group on the node, enter the following command:
     ```
     rdqmadm -s
     ```
     To delete the HA group from the node, enter the following command:
     ```
     rdqmadm -u
     ```
  2. Run the script *MQ_INSTALLATION_PATH*`/samp/rdqm/firewalld/unconfigure.sh` to undo the firewall configuration.
  3. Run the script `uninstallRDQMsupport` to uninstall IBM MQ, RDQM support, DRBD, and Pacemaker.
- V 9.0.5 To uninstall DR RDMQ support:
  1. Back up the queue manager running on the primary node, see Backing up and restoring IBM queue manager data.
  2. Delete the queue manager on both the primary and secondary nodes, see Deleting a DR RDQM.
  3. Run the script `uninstallRDQMsupport` to uninstall IBM MQ, RDQM support, DRBD, and Pacemaker.

Related information:

rdqmadm (administer replicated data queue manager cluster)

Installing IBM MQ for z/OS

z/OS

Installation tasks that are associated with installing IBM MQ on z/OS systems are grouped in this section.

About this task

IBM MQ for z/OS uses the standard z/OS installation procedure. It is supplied with a Program Directory that contains specific instructions for installing the program on a z/OS system. You must follow the instructions in the appropriate Program Directory, which can be downloaded from the IBM Publications Center:

- LTS *Program directory for IBM MQ for z/OS Long Term Support Release V9.0.0 (GI13-3386)*
- CD *Program directory for IBM MQ for z/OS Continuous Delivery Release V9.0.x (GI13-3391)*

The Program Directory includes not only details of the installation process, but also information about the prerequisite products and their service or maintenance levels.

SMP/E, used for installation on the z/OS platform, validates the service levels and prerequisite and corequisite products, and maintains the SMP/E history records to record the installation of IBM MQ for z/OS. It loads the IBM MQ for z/OS libraries and checks that the loads have been successful. You then have to customize the product to your own requirements.

Before you install and customize IBM MQ for z/OS, you must decide the following:

- Whether you are going to install one of the optional national language features. See National language support.
- Which communications protocol and distributed queuing facility you are going to use. See Communications protocol and distributed queuing.
- What your naming convention for IBM MQ objects will be. See Naming conventions.
- What command prefix string (CPF) you are going to use for each queue manager. See Using command prefix strings.

- CD When upgrading from a previous Continuous Delivery release through the installation of PTFs, decide whether any USERMODs that have been applied to IBM MQ for z/OS will still be required. Remove the USERMODs before installation of the Continuous Delivery PTFs, or use the SMP/E BYPASS(ID) option on APPLY. If neither of these actions is performed, an `SMP/E MODID ERROR GIM38201E` will be received.

  The PTFs for the latest Continuous Delivery release can be determined by using SMP/E FIXCAT HOLDDATA category IBM.MQ.V9R0M*n*, where *n* is the modification level. For example, category IBM.MQ.V9R0M2 identifies fixes that upgrade IBM MQ for z/OS Version 9.0 Continuous Delivery to modification level 2.

You also need to plan how much storage you require in your z/OS system to accommodate IBM MQ; Planning your storage and performance requirements on z/OS helps you plan the amount of storage required.

Procedure

1. Check that your system hardware, and software levels meet the minimum requirements. See "Checking requirements on z/OS" on page 550.
2. Plan your installation See "Planning to install IBM MQ for z/OS" on page 550.
3. Install and configure IBM MQ for z/OS, by following the instructions detailed in the Program Directory. See also the information in the subtopics for further guidance.

z/OS installation overview

IBM MQ functions are provided as a number of different products, which are installed together to provide the capability required.

The different products are:

IBM MQ for z/OS
Provides IBM MQ capability, connectivity on and off the z/OS platform, and excellent integration with z/OS software, such as CICS, IMS, WebSphere Application Server, and Db2. Licensed under a Monthly License Charge (MLC) model.

IBM MQ for z/OS Value Unit Edition (VUE)
Same functionality as IBM MQ for z/OS, but intended for new workloads running within a zNALC LPAR only. Licensed under a One Time Charge (OTC) model. Can coexist and interact with IBM MQ MLC offerings in other LPARs.

Note, that from an installation perspective, the same FMIDs as for IBM MQ for z/OS are installed, then, an additional enablement feature is added, which changes product usage recording for billing purposes.

IBM MQ Managed File Transfer for z/OS (MFT)
Equivalent function as Managed File Transfer on Multiplatforms, though more closely integrated with base IBM MQ offering as of Version 8. Must be locally bound to a z/OS queue manager.

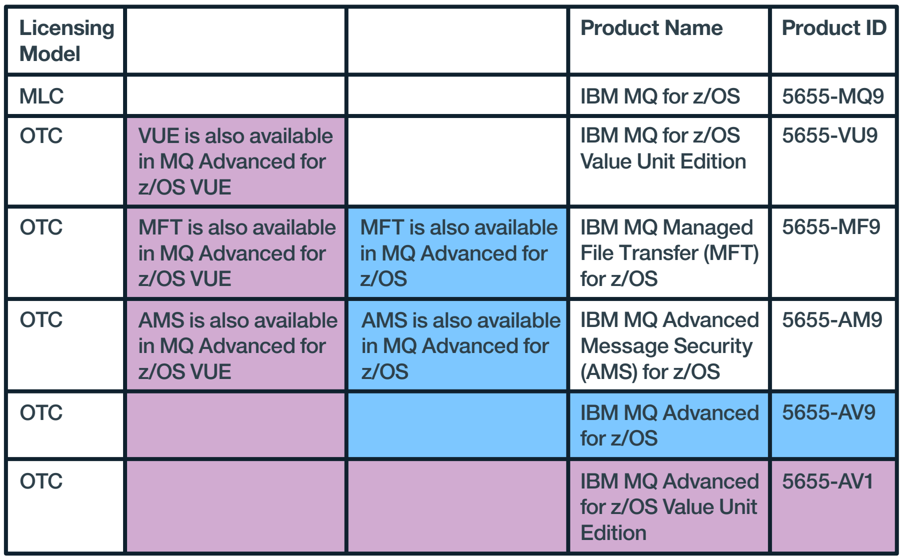
IBM MQ Advanced Message Security for z/OS (AMS)
Provides end to end encryption of messages throughout the IBM MQ network. Data is encrypted at rest, as well as when data is being transmitted.

Note, that from an installation perspective, AMS only provides an enablement feature, which permits encryption code integrated into the queue manager to be used.

IBM MQ Advanced for z/OS
Bundling of IBM MQ Managed File Transfer for z/OS and IBM MQ Advanced Message Security for z/OS only; that is, no IBM MQ for z/OS. Can be deployed with IBM MQ MLC or VUE offerings

IBM MQ Advanced for z/OS, Value Unit Edition
Bundling of IBM MQ for z/OS Value Unit Edition, together with IBM MQ Managed File Transfer for z/OS, and IBM MQ Advanced Message Security for z/OS

The two different licensing models, together with their associated bundles, are shown in the following table:



Licensing Model			Product Name	Product ID
MLC			IBM MQ for z/OS	5655-MQ9
OTC	VUE is also available in MQ Advanced for z/OS VUE		IBM MQ for z/OS Value Unit Edition	5655-VU9
OTC	MFT is also available in MQ Advanced for z/OS VUE	MFT is also available in MQ Advanced for z/OS	IBM MQ Managed File Transfer (MFT) for z/OS	5655-MF9
OTC	AMS is also available in MQ Advanced for z/OS VUE	AMS is also available in MQ Advanced for z/OS	IBM MQ Advanced Message Security (AMS) for z/OS	5655-AM9
OTC			IBM MQ Advanced for z/OS	5655-AV9
OTC			IBM MQ Advanced for z/OS Value Unit Edition	5655-AV1

Figure 58. IBM MQ for z/OS product bundles

Both the IBM MQ Advanced Message Security product, and IBM MQ for z/OS Value Unit Edition product, provide their own module that only enables their respective function. The functional code is integrated into the base IBM MQ for z/OS code for efficiency, and supplied and serviced through the base IBM MQ for z/OS code.

V 9.0.3 From IBM MQ Version 9.0.3, rather than separately install these enablement modules, a runtime configuration option is available. This provides simpler and more granular control of which functions are available, and better recording of the capabilities used in SMF billing data at an individual queue manager level. See "Product usage recording with IBM MQ for z/OS products" on page 555 for more information.

Program directories provide instructions for SMP/E installation of the program materials on to a target system. The Customizing IBM MQ for z/OS topics guide you through customization of the code, and creating customized execution units, for example the queue manager and file transfer agents.

Related tasks:

Installing Advanced Message Security
Use the information for your platform to guide you through installing the Advanced Message Security (AMS) component.

Related information:

Maintaining and migrating

Managed File Transfer product options

Checking requirements on z/OS

z/OS

Before you install IBM MQ on z/OS, you must check for the latest information and system requirements.

About this task

A summary of the tasks that you must complete to check system requirements is listed here with links to further information.

Procedure

1. Check that you have the latest information, including information on hardware and software requirements. See "Where to find product requirements and support information" on page 195.
2. Check that your systems meet the initial hardware and software requirements for z/OS. Before attempting to install, and run IBM MQ for z/OS, ensure that your system hardware, and software levels meet the minimum requirement. You can check the minimum required levels at IBM MQ for z/OS requirements.

   Note that the System Requirements page for IBM MQ Version 9.0 uses the Software Product Compatibility Reports (SPCR) tool. The SPCR tool enables you to go directly to the:
   - Supported operating systems
   - Prerequisites
   - System requirements, and
   - Optional supported software.
3. Check that you have the correct licenses. See "License requirements" on page 194 and IBM MQ license information.

Planning to install IBM MQ for z/OS

z/OS

To install the IBM MQ product your hardware, and software environment must meet minimum requirement levels. You must also consider the national language features, communications protocols, and naming conventions to be used.

National language support

You can choose one of the following national languages for the IBM MQ operator messages and the IBM MQ operations and control panels (including the character sets used). Each language is identified by one of the following language letters:

C	Simplified Chinese
E	U.S. English (mixed case)
F	French

K	Japanese
U	U.S. English (uppercase)

The samples, IBM MQ commands, and utility control statements are available only in mixed case U.S. English.

Communications protocol and distributed queuing

The distributed queuing facility provided with the base product feature of IBM MQ can either use APPC (LU 6.2), TCP/IP from IBM, or any TCP product which supports the z/OS Unix Sockets API. The distributed queuing facility is also known as the channel initiator and the mover.

You must perform the following tasks to enable distributed queuing:
- Choose which communications interface to use. This can be either, or both, of the following:
  - APPC (LU 6.2)
  - TCP/IP
- Customize the distributed queuing facility and define the IBM MQ objects required.
- Define access security.
- Set up your communications. This includes setting up your TCPIP.DATA data set if you are using TCP/IP, LU names, and side information if you are using APPC. This is described in Setting up communication for z/OS.

Naming conventions

It is advisable to establish a set of naming conventions when planning your IBM MQ systems. The names you choose will probably be used on different platforms, so you should follow the convention for IBM MQ, not for the particular platform.

IBM MQ allows both uppercase and lowercase letters in names, and the names are case sensitive. However, some z/OS consoles fold names to uppercase, so do not use lowercase letters for names unless you are sure that this will not happen.

You can also use numeric characters and the period (.), forward slash (/), underscore (_) and percent (%) characters. The percent sign is a special character to Security Server (previously known as RACF®), so do not use it in names if you are using Security Server as your External Security Manager. Do not use leading or trailing underscore characters if you are planning to use the Operations and Control panels.

For more information, see Rules for naming IBM MQ objects.

Choosing names for queue managers and queue-sharing groups

Each queue manager and queue-sharing group within a network must have a unique name. Do not use the same name for a queue manager and a queue-sharing group. On z/OS the names of queue managers and queue-sharing groups can be up to four characters long. Each Db2 system and data-sharing group within the network must also have a unique name.

The names of queue manager and queue-sharing groups can use only uppercase alphabetic characters, numeric characters, and dollar sign ($), number sign (#) or at sign (@); they must not start with a numeric character. Queue-sharing group names that are less than four characters long are padded internally with at signs, so do not use names ending in the at sign.

The queue manager name is the same as the z/OS subsystem name. You might identify each subsystem as a queue manager by giving it the name `QM` *xx* (where *xx* is a unique identifier), or you might choose a naming convention like `ADDX`, where `A` signifies the geographic area, `DD` signifies the company division, and `X` is a unique identifier.

You might want to use your naming convention to distinguish between queue managers and queue-sharing groups. For example, you might identify each queue-sharing group by giving it the name `QG` *`xx`* (where *`xx`* is the unique identifier).

Choosing names for objects

Queues, processes, name lists, and clusters can have names up to 48 characters long. Channels can have names up to 20 characters long and storage classes can have names up to 8 characters long.

If possible, choose meaningful names within any constraints of your local conventions. Any structure or hierarchy within names is ignored by IBM MQ, however, hierarchical names can be useful for system management. You can also specify a description of the object when you define it to give more information about its purpose.

Each object must have a unique name within its object type. However, each object type has a separate namespace, so you can define objects of different types with the same name. For example, if a queue has an associated process definition, it is a good idea to give the queue and the process the same name. It is also a good idea to give a transmission queue the same name as its destination queue manager.

You could also use the naming convention to identify whether the object definition is private or a global. For example, you could call a namelist `project_group.global` to indicate that the definition is stored on the shared repository.

Application queues

Choosing names that describe the function of each queue helps you to manage these queues more easily. For example, you might call a queue for inquiries about the company payroll `payroll_inquiry`. The reply-to queue for responses to the inquiries might be called `payroll_inquiry_reply`.

You can use a prefix to group related queues. This means that you can specify groups of queues for administration tasks like managing security and using the dead-letter queue handler. For example, all the queues that belong to the payroll application might be prefixed by `payroll_`. You can then define a single security profile to protect all queues with names beginning with this prefix.

You can also use your naming convention to indicate that a queue is a shared queue. For example, if the payroll inquiry queue was a shared queue, you might call it `payroll_inquiry.shared`.

Storage classes and coupling facility structures

The character set you can use when naming storage classes and coupling facility structures is limited to uppercase alphabetic and numeric characters. You should be systematic when choosing names for these objects.

Storage class names can be up to 8 characters long, and must begin with an alphabetic character. You will probably not define many storage classes, so a simple name is sufficient. For example, a storage class for IMS bridge queues could be called `IMS`.

Coupling facility structure names can be up to 12 characters long, and must begin with an alphabetic character. You could use the name to indicate something about the shared queues associated with the coupling facility structure (that they all belong to one suite of applications for example). Remember that in the coupling facility, the structure names are the IBM MQ name prefixed by the name of the queue-sharing group (padded to four characters with @ symbols).

Choosing names for channels

To help you manage channels, it is a good idea if the channel name includes the names of the source and target queue managers. For example, a channel transmitting messages from a queue manager called QM27 to a queue manager called QM11 might be called `QM27/QM11`.

If your network supports both TCP and SNA, you might also want to include the transport type in the channel name, for example `QM27/QM11_TCP`. You could also indicate whether the channel is a shared channel, for example `QM27/QM11_TCP.shared`.

Remember that channel names cannot be longer than 20 characters. If you are communicating with a queue manager on a different platform, where the name of the queue manager might contain more than 4 characters, you might not be able to include the whole name in the name of the channel.

Using command prefix strings

Each instance of IBM MQ that you install must have its own *command prefix* string (CPF). You use the CPF to identify the z/OS subsystem that commands are intended for. It also identifies the z/OS subsystem from which messages sent to the console originate.

You can issue all MQSC commands from an authorized console by inserting the CPF before the command. If you enter commands through the system command input queue (for example, using CSQUTIL), or use the IBM MQ operations and control panels, you do not use the CPF.

To start a subsystem called CSQ1 with CPF that is ' `+CSQ1` ', issue the command `+CSQ1 START QMGR` from the operator console (the space between the CPF and the command is optional).

The CPF also identifies the subsystem that is returning operator messages. The following example shows `+CSQ1` as the CPF between the message number and the message text.

```
CSQ9022I +CSQ1 CSQNCDSP ' DISPLAY CMDSERV' NORMAL COMPLETION
```

See Defining command prefix strings (CPFs) for information about defining command prefix strings.

Delivery media

z/OS

IBM MQ for z/OS is supplied on 3590 cartridge only.

One cartridge contains the product code together with the following language features; U.S. English (mixed case), U.S. English (uppercase), French, Chinese, and Japanese.

Customizing IBM MQ and its adapters

z/OS

IBM MQ requires some customization after installation to meet the individual and special requirements of your system, and to use your system resources in the most effective way.

For a list of tasks that you must perform when you customize your system, see Customizing IBM MQ for z/OS.

Using queue-sharing groups

If you want to use queue-sharing groups, you do not have to set them up when you install IBM MQ, you can do this at any time.

For details of how to manage your queue-sharing groups when you have set them up, see Managing queue-sharing groups.

Verifying your installation of IBM MQ for z/OS

z/OS

After the installation and customization has been completed, you can use the installation verification programs (IVPs) supplied with IBM MQ for z/OS to verify that the installation has been completed successfully.

The IVPs supplied are assembler language programs and you should run them after you have customized IBM MQ for z/OS to suit your needs. They are described in Running the basic installation verification program.

Macros intended for customer use

z/OS

The macros identified in this topic are provided as programming interfaces for customers in support of features that are specific to IBM MQ for z/OS.

The 'C' include files, COBOL copy files, PL/I include files and assembler macros that are provided as programming interfaces for customers in support of features that apply across many IBM MQ platforms are described in the Constants.

Note: Do not use as programming interfaces any IBM MQ macros other than those interfaces identified in this topic or in the Constants

General-use programming interface macros

The following assembler macros are provided to enable you to write programs that use the services of IBM MQ. The macros are supplied in library thlqual.SCSQMACS.

- CMQXCALA
- CMQXCFBA
- CMQXCFCA
- CMQXCFLA
- CMQXCDFA
- CMQXCINA
- CMQXCVCA

Product-sensitive programming interface macros

The following assembler macros are provided to enable you to write programs that use the services of IBM MQ. The macros are supplied in library thlqual.SCSQMACS. Product-sensitive interfaces are open to change between different releases of the product.

- CSQBDEF
- CSQDQEST
- CSQDQIST
- CSQDQJST
- CSQDQLST
- CSQDQMAC
- CSQDQMST
- CSQDQPST
- CSQDQSST

- CSQDQWHC
- CSQDQWHS
- CSQDQ5ST
- CSQDWQ
- CSQDWTAS
- CSQQDEFX
- CSQQLITX

Product usage recording with IBM MQ for z/OS products

z/OS

To determine the product usage, the z/OS system records the amount of processor time that is used by the product when it processes.

z/OS can measure how much processing time is spent in doing work on behalf of the IBM MQ queue manager that is handling MQI calls, executing MQSC commands, or performing some other action to support the messaging and queuing functions that are used by your application programs. The amount of processing time is recorded in a file at hourly intervals, and the hourly records are totaled at the end of a month. In this way, the total amount of time used by the IBM MQ for z/OS product on your behalf is computed, and used to determine how much you pay for your use of the IBM MQ for z/OS product that month.

Product usage recording is implemented as follows:

- When IBM MQ for z/OS is installed, it identifies itself to z/OS and requests that the *System Management Facilities (SMF)* mechanism within z/OS is to automatically measure how much processor time is used by the IBM MQ for z/OS product.
- V 9.0.3 The default product used by registration must be overridden at run-time for particular software execution instances which are processing under the license provided by a particular software bundle.

  For example, if you are licensed to use IBM MQ Advanced for z/OS, Value Unit Edition (VUE), then:

  - Queue manager
  - Advanced message security, and
  - Managed file transfer

  instances must be identified as using the ADVANCEDVUE product identifier, so that month end usage reports correctly identify the product used.
- When enabled, the z/OS usage measurement facility collects usage figures for each hour of the day, and generates usage records that are added to a report file on disk.
- At the end of one full month, these usage records are collected by a program, which generates a report of product usage for the month. This report is used to determine the charge for the IBM MQ for z/OS product.

For more information about product usage recording and the Sub-Capacity Reporting Tool (SCRT), see IBM Z Software Pricing. For information about the MULCCAPT parameter see, Using CSQ6SYSP.

V 9.0.3

Over-riding the default product associated with usage

IBM MQ Version 9.0.3 introduces an improved method for associating IBM MQ usage with the licensed Product ID (PID) so that workload reporting tools, for example, SCRT and MWRT, correctly reflect usage.

Each of the following products uses a different PID:
- Base IBM MQ
- Advanced Message Security
- Managed File Transfer
- IBM MQ for z/OS Value Unit Edition (VUE)
- IBM MQ Advanced for z/OS, Value Unit Edition

There are two alternative mechanisms to associate the correct PID with a running instance of IBM MQ:
1. Most useful for a long term deployment of IBM MQ - The CSQ6USGP macro selects the correct PID, and is built into the queue manager zPARM parameter.
2. Most useful for a test system which needs to run for short times under different PIDs - Parameters on the START QMGR command select the PIDs. These run-time parameters, **QMGRPROD** and **AMSPROD**, can be entered on the command, or coded into the MSTR JCL.

   Note that a value entered on a START command overrides any values encoded into ZPARM.

In all cases, the selected PIDs are shown in startup messages CSQY036I and CSQ0619I.

For Managed File Transfer, the PID is selected using an fteSetProductId command.

If no value is set by the preceding mechanisms, a default PID is used.

Sample scenarios

You have MLC IBM MQ for z/OS and have purchased Advanced Message Security
: No changes are necessary, the default PIDs are correctly recorded, 5665-MQ9 for IBM MQ use and 5665-AM9 for Advanced Message Security usage.

You are migrating a single queue manager from MLC to VUE
: Use either the ZPARM or START QMGR mechanism, to select **QMGRPROD**=*VUE*

You have installed a new IBM MQ Advanced for z/OS, Value Unit Edition on a new LPAR
: Use either the ZPARM or START QMGR mechanisms to select both **QMGRPROD**=*ADVANCEDVUE* and **AMSPROD**=*ADVANCEDVUE*.

  For MFT deployments use **fteSetProductID ADVANCEDVUE**

IBM MQ for z/OS Value Unit Edition (VUE)

z/OS

IBM MQ for z/OS Value Unit Edition (VUE) provides all the function and capability of the base IBM MQ for z/OS, in a format that offers a one-time-charge (OTC) price metric for eligible workloads that are deployed in qualified IBM Z New Application License Charge (zNALC) logical partitions (LPARs).

The term, Eligible Workload, is defined as new workload that executes using the IBM MQ for z/OS VUE server environment, on condition that the workload is qualified and approved through the zNALC qualification process.

For further information on zNALC, see IBM Z Software Pricing

The OTC price metric provides an alternative pricing model for new IBM MQ for z/OS-connected applications and new IBM MQ for z/OS VUE service enablement workloads.

Support for the zNALC metric offers a reduced price for the z/OS operating system on LPARs that run a qualified application.

IBM MQ for z/OS VUE can connect to other supported versions of IBM MQ for z/OS (whether in zNALC or non-zNALC environments) for workload federation and systems management.

IBM MQ for z/OS VUE allows connections from IBM MQ clients, that run on other platforms.

Installing VUE

An order for VUE is fulfilled by delivery of two products:
- IBM MQ for z/OS (5655-MQ9), either Long Term Support (LTS) release or Continuous Delivery (CD) release.
- The VUE enabling product, IBM MQ for z/OS Value Unit Edition Version 9.0 (5655-VU9).

Note: The VUE enabling product enables either IBM MQ Version 9.0 LTS release or IBM MQ Version 9.0 CD release to conform with the licensing requirements of Value Unit Edition operation.
The products are separately installed using SMP/E following the process documented in their respective Program Directories which can be downloaded from the IBM Publications Center:

- LTS *Program directory for IBM MQ for z/OS Long Term Support Release V9.0.0 (GI13-3386)*
- LTS *Program directory for IBM MQ for z/OS Value Unit Edition Long Term Support Release V9.0.0 (GI13-3387)*
- CD *Program directory for IBM MQ for z/OS Continuous Delivery Release V9.0.x (GI13-3391)*
- CD *Program directory for IBM MQ for z/OS Value Unit Edition Continuous Delivery Release V9.0.x (GI13-3395)*

.

Enabling VUE

To enable a queue manager to run an eligible workload in a zNALC LPAR, the SCUEAUTH library created by installation of the VUE enabling product must be added to the STEPLIB concatenation of the xxxxMSTR procedure for that queue manager:
- The SCUEAUTH library should be APF authorized
- The SCUEAUTH libary must be concatenated ahead of the SCSQAUTH library,

For example, the CSQ4MSTR sample would be modified as follows:

```
//PROCSTEP EXEC PGM=CSQYASCP,REGION=0M,MEMLIMIT=2G
//*
//STEPLIB  DD DSN=hlq.SCSQANLE,DISP=SHR
//     DD DSN=hlq.SCUEAUTH,DISP=SHR
//     DD DSN=hlq.SCSQAUTH,DISP=SHR
-
```

When SCUEAUTH is added to the STEPLIB concatenation, the characters VUE appear in the release level shown by message CSQY000I during queue manager startup. The queue manager starts only in an LPAR configured for zNALC workload by name or IPL parameter.

Characteristics of a VUE-enabled queue manager

A VUE-enabled queue manager has all the function and capability of the base queue manager. Additionally, clients will be enabled during channel initiator startup.

A VUE-enabled queue manager records usage information in SMF89 records with the product name and identifier for IBM MQ for z/OS Value Unit Edition (VUE) instead of those for the IBM MQ product.

A VUE-enabled queue manager can:

- Connect to other queue managers and clients in a network, according to the connectivity capabilities of the base queue manager installation.
- Participate in a queue-sharing group with other queue managers provided the base queue manager versions are able to interoperate, regardless of whether other members are standard or VUE function queue managers.

Installing Managed File Transfer for z/OS

z/OS

You install Managed File Transfer on your IBM MQ for z/OS system by using SMP/E.

About this task

Managed File Transfer for z/OS uses the standard z/OS installation procedure. It is supplied with a Program Directory that contains specific instructions for installing the program. You must follow the instructions in the appropriate Program Directory, which can be downloaded from the IBM Publications Center:

- LTS *Program directory for Managed File Transfer for z/OS Long Term Support Release V9.0.0 (GI13-3389)*
- CD *Program directory for Managed File Transfer for z/OS Continuous Delivery Release V9.0.x (GI13-3392)*

The instructions in the Program Directory include not only details of the installation process, but also information about the prerequisite products and their service or maintenance levels.

SMP/E, used for installation on the z/OS platform, validates the service levels and prerequisite and corequisite products, and maintains the SMP/E history records to record the installation of Managed File Transfer. The process loads the appropriate libraries and checks that the loads have been successful. You then have to customize the product to your own requirements.

Note: For Version 9.0, the supported version of Java for Managed File Transfer for z/OS is Java 1.8.

Procedure

1. Plan your installation. See Planning for Managed File Transfer for items you need to consider before installing the component.
2. Install the product by following the instructions detailed in the Program Directory.
3. Check that the SMP/E installation process has created the product JCL library USERID.MFTV800.SBFGCMDS. If this JCL library has not been created during the installation process, create the library and submit the job USERID.ZOS.JCL(COPYJCL1).

What to do next

When you have installed the product, you must carry out some customization tasks. For more information, see Configuring Managed File Transfer for z/OS.

Related information:
Managed File Transfer for z/OS
Planning for Managed File Transfer

Installing Advanced Message Security on z/OS

z/OS

You can install Advanced Message Security component on z/OS by using SMP/E.

About this task

Advanced Message Security for z/OS (AMS) is a separately licensed enabling product that extends IBM MQ to provide a high level of protection for sensitive data flowing through the IBM MQ network using a public key cryptography model.

Advanced Message Security for z/OS is installed separately using SMP/E by following the process documented in the Program Directory, which can be downloaded from the IBM Publications Center:

- LTS Program directory for Advanced Message Security for z/OS Long Term Support Release V9.0.0 (GI13-3388)
- CD Program directory for Advanced Message Security for z/OS Continuous Delivery Release V9.0.x (GI13-3398)

When you have completed the SMP/E installation, it provides the SDRQAUTH library which contains the Advanced Message Security for z/OS enablement module. You must make the enablement module available for processing during queue manager startup, either by adding to the system linklist or LPA, or for individual queue managers, by including in the STEPLIB concatenation.

The enablement module can be used with either a Long Term Support release or Continuous Delivery release of IBM MQ for z/OS to activate the Advanced Message Security for z/OS functions.

Procedure

1. Install Advanced Message Security for z/OS using SMP/E. When installing Advanced Message Security for z/OS, you must follow the instructions in the appropriate Program Directory.
2. Enable Advanced Message Security for z/OS separately for each queue manager. Completing the additional customization tasks described in Customizing IBM MQ for z/OS.

   The following tasks are relevant when adding AMS support to a queue manager:
   - Task 2: APF authorize the IBM MQ load libraries
   - Task 3: Update the z/OS link list and LPA
   - Task 4: Update the z/OS program properties table
   - Task 13: Customize the initialization input data sets
   - Task 17: Tailor your system parameter module
     - Using CSQ6SYSP
   - Task 23: Create procedures for Advanced Message Security
   - Task 24: Set up the started task user Advanced Message Security
   - Task 25: Grant RACDCERT permissions to the security administrator for Advanced Message Security
   - Task 26: Grant users resource permissions for Advanced Message Security

   You also need to configure certificates and policies, which are described in
   - Using certificates on z/OS

- Security policies
- Example configurations on z/OS

Results

Advanced Message Security component has been installed successfully.

Related information:

Advanced Message Security

Advanced Message Security for z/OS

Advanced Message Security for z/OS (AMS) is a separately licensed enabling product that extends IBM MQ to provide a high level of protection for sensitive data flowing through the IBM MQ network using a public key cryptography model.

This information has moved. See "Installing Advanced Message Security on z/OS" on page 559.

Related information:

Advanced Message Security

Installing IBM MQ Advanced for z/OS

z/OS V 9.0.0 MQ Adv.

Use this topic to understand how you install IBM MQ Advanced for z/OS on your system.

About this task

IBM MQ Advanced for z/OS is a bundling of the Advanced Message Security for z/OS and Managed File Transfer for z/OS products.

Procedure

For installation instructions, follow the guidance in "Installing Advanced Message Security on z/OS" on page 559 and "Installing Managed File Transfer for z/OS" on page 558. The Program Directories for IBM MQ Advanced for z/OS can be downloaded from the IBM Publications Center:

- LTS *Program directory for IBM MQ Advanced for z/OS Long Term Support Release V9.0.0 (GI13-3390)*
- CD *Program directory for IBM MQ Advanced for z/OS Continuous Delivery Release V9.0.x (GI13-3396)*

Installing IBM MQ Advanced for z/OS, Value Unit Edition

z/OS V 9.0.0 MQ Adv. VUE

Use this topic to understand how you install IBM MQ Advanced for z/OS, Value Unit Edition (VUE) on your system.

About this task

IBM MQ Advanced for z/OS, VUE is a bundling of the Advanced Message Security for z/OS, Managed File Transfer for z/OS, and IBM MQ for z/OS Value Unit Edition (VUE) products.

Procedure

For installation instructions, follow the guidance in “Installing Advanced Message Security on z/OS” on page 559, “Installing Managed File Transfer for z/OS” on page 558, and “IBM MQ for z/OS Value Unit Edition (VUE)” on page 556.

CD The *Program directory for IBM MQ Advanced for z/OS Value Unit Edition Continuous Delivery Release V9.0.x (GI13-3397)* can be downloaded from the IBM Publications Center.

Related tasks:

“Installing IBM MQ Advanced for Multiplatforms” on page 519
Installation tasks associated with IBM MQ Advanced for Multiplatforms are grouped in this section.

Related information:

V 9.0.5 DISPLAY QMGR ADVCAP

V 9.0.5 MQCMD_INQUIRE_Q_MGR MQIA_ADVANCED_CAPABILITY

Installing IBM MQ into IBM Cloud Private

Linux V 9.0.3 MQ Adv.

Add an IBM MQ image into an IBM Cloud Private cluster, then deploy a queue manager into IBM Cloud Private.

About this task

IBM Cloud Private offers managed container services in the cloud. You can run IBM MQ in an IBM Cloud Private container.

Procedure

1. Add an IBM MQ image into an IBM Cloud Private cluster.
2. Deploy a queue manager into IBM Cloud Private.

Adding an IBM MQ image into an IBM Cloud Private cluster

Linux V 9.0.3 MQ Adv.

Prepare your IBM Cloud Private cluster to deploy a production-ready image for IBM MQ.

About this task

IBM Cloud Private offers managed container services in the cloud. You can download an IBM MQ image from Passport Advantage and import it into an IBM Cloud Private container.

Procedure

1. Download the latest IBM MQ image from Passport Advantage.

   For details of available downloads, go to Downloading IBM MQ Version 9.0 then click the tab for the release that you want to download. The name and number of the part to download are listed in a table. For example, for IBM MQ Version 9.0.4 the part is *IBM MQ Advanced V9.0.4 Continuous Delivery Release for IBM Cloud Private 2.1*, part number *CNMV6ML*.
2. Import the downloaded archive file into IBM Cloud Private.

   See Installing bundled products in the IBM Cloud Private product documentation.

What to do next

You are now ready to Deploy a queue manager into IBM Cloud Private.

Deploying a queue manager into IBM Cloud Private

Linux V 9.0.3 MQ Adv.

Use the IBM Cloud Private management console to deploy a queue manager into IBM Cloud Private.

Before you begin

This task assumes that you have already added an IBM MQ image into an IBM Cloud Private cluster.

About this task

IBM Cloud Private offers managed container services in the cloud. After you have added an IBM MQ image into an IBM Cloud Private container, you can use the IBM Cloud Private management console to deploy a queue manager.

Procedure

1. Determine which *Storage Class* to use.

   IBM MQ stores queues, messages and configuration data on a *Persistent Volume*. A *Persistent Volume* can be created for you automatically during deployment, if you specify a *Storage Class* (for example, `silver` or `gold`).

   To check which storage classes are available in your IBM Cloud Private cluster, use the command **`kubectl get storageclasses`**. If no storage classes are defined, create a persistent volume manually. See Storage in the IBM Cloud Private product documentation.

   The default volume size for IBM MQ is 2 GB.
2. Open the IBM Cloud Private management console in a web browser, and click **Menu** > **Catalog**.

   See Accessing your IBM Cloud Private cluster by using the management console in the IBM Cloud Private product documentation.
3. Select the `ibm-mqadvanced-server-prod` chart from the list.

4. Select **Configure**, then complete the following configuration steps:
   a. Enter a release name.
   b. Read and accept the license agreements.
   c. Under the dataPVC section, set **storageclass** to the storage class name determined earlier. If your cluster does not use storage classes, set **useDynamicProvisioning** to `false`.
   d. Under the image section, set the repository to the full image path.

      For example `mycluster.icp:8500/default/ibm-mqadvanced-server-prod`
   e. Under the image section, set the tag to the image tag.

      For example `9.0.4-20171106-1625-x86_64`
   f. Optional: If you need a Kubernetes pull secret to access the image registry, add it as the **pullSecret**.
   g. Optional: Under the queueManager section, set the name of the queue manager.
5. Click **Install** to deploy your queue manager as a *Helm release*.

Maintaining and migrating IBM MQ

Maintenance, upgrade, and migration have three distinct meanings for IBM MQ. The definitions are described here. The following sections describe the various concepts associated with migration, followed by the various tasks needed; these tasks are platform-specific where needed.

About this task

Attention: The information in this section applies to both Continuous Delivery (CD) and Long Term Support (LTS) releases.

Any information that applies specifically to an LTS or CD release is marked with the appropriate icon.

IBM MQ uses the terms *maintenance*, *upgrade* and *migration* as follows:

Maintenance is the application of a fix pack, interim fix or Program Temporary Fix (PTF).
Maintenance has one main characteristic. Those fixes, whether they are applied using a maintenance installation tool, or installed using a manufacturing refresh on top of an installation, are at the same command level as the existing code. No migration is required after applying maintenance. The installation can be restored to its previous level and any changed queue managers or applications will continue to work at the restored code level. However, you should test applications with the new level of IBM MQ code.

For more information, see "Applying maintenance to IBM MQ" on page 570.

Upgrading is the process of taking an existing IBM MQ installation and upgrading to a new level of code.
Unless you are upgrading the fix level of IBM MQ, but not its command level, an upgrade must be followed by migration. Upgrades can be backed out, as long as no migration has taken place. The process of removing an upgrade varies by platform and how the upgrade was applied. Upgrades that change the command level of IBM MQ require queue manager migration before applications can reconnect.

For more information, see "Applying upgrades and fixes to IBM MQ" on page 618.

Migration is the process of updating queue manager data to match a newer level of code.
Migration occurs the first time a queue manager is started with the newer level of code, and always follows an upgrade that changes the queue manager command level, both automatic and manual changes. Migration is the transformation of queue manager data, applications, and the environment that the queue manager runs in. Once migration has occurred, the queue manager can no longer be started by an earlier code level. On most platforms, queue manager migration is not reversible:

- Multi Migration cannot be reversed on IBM MQ for Multiplatforms. This restriction applies whether your enterprise uses the Long Term Support (LTS) release or Continuous Delivery (CD) release model.
- z/OS From IBM MQ for z/OS Version 9.0, you can backwards migrate queue managers only if you are using the LTS release. For more information, see IBM MQ release types.

For more information, see "Migrating IBM MQ" on page 625.

Related information:
IBM MQ for z/OS Program directory PDF files

Where to find more information about maintaining and migrating

Where to look for more information, for example if you are getting started with migrating and maintaining IBM MQ.

Getting started with maintaining and migrating IBM MQ

If you are not familiar with IBM MQ migration, start by reading the following information:
- The "Migration concepts and methods" on page 645 section: use these topics to find out more about the concepts that you must understand before planning migration tasks, including the difference between maintenance, migration, and upgrading and which migration paths are supported.
- The "IBM WebSphere MQ / IBM MQ Migration Guides" on page 567: use these guides to find more information about planning the migration process for your release and platform.

New features and changes in this release

For information about new features and changes in this release, see the following topics:
- V 9.0.0 What's new and changed in IBM MQ Version 9.0
- CD What's new and changed in Version 9.0.x Continuous Delivery
- LTS What's new and changed in Version 9.0.0.x Long Term Support

New features and changes in earlier releases

Some new features and changes from earlier releases might have an impact on planning your migration because they affect the behavior of existing applications or the automation of management tasks. For information on where to find details of these changes in the product documentation for earlier releases, see What was new and changed in earlier releases.

Important: If you are migrating your system from a version of IBM WebSphere MQ before Version 7.0, you must migrate your system to Version 7.0.1, or Version 7.1 before you migrate to the latest version. See the appropriate version of the product documentation for information on how to carry out the task. For links to earlier versions of the product documentation not available in IBM Knowledge Center, see the IBM MQ documentation library web page.

System requirements and prerequisites

From IBM MQ Version 8.0, you can use the Software Product Compatibility Reports (SPCR) tool to find information on supported operating systems, system requirements, prerequisites, and optional supported software. For more information about the SPCR tool and links to reports for each supported platform, see the System Requirements for IBM MQ Version 9.0 web page.

For links to system requirements information for all releases of IBM WebSphere MQ or IBM MQ, see System Requirements for IBM MQ.

For information about limitations and known problems for IBM MQ Version 9.0 and its maintenance, see the product readme file, which is available from the IBM MQ, WebSphere MQ, and MQSeries product readmes web page.

IBM WebSphere MQ / IBM MQ Migration Guides

Multi The *IBM WebSphere MQ / IBM MQ Migration Guide* provides information to help you plan the process of migrating from an older version to a new version of IBM MQ for Multiplatforms.

- For an introduction to the guide and its contents, see the developerWorks® blog article IBM WebSphere MQ / IBM MQ Migration Guide.
- To view the guide in your web browser, click the following link: IBM WebSphere MQ / IBM MQ Migration Guide - HTML version.
- To download the guide as a PDF file, click the following link: IBM WebSphere MQ / IBM MQ Migration Guide - PDF file.

z/OS The *IBM WebSphere MQ/ IBM MQ for z/OS Migration Guide* provides information to help you plan the process of migrating from an older version to a new version of the product on z/OS.

- For an introduction to the guide and its contents, see the developerWorks blog article IBM WebSphere MQ / IBM MQ for z/OS Migration Guide.
- To view the guide in your web browser, click the following link: IBM WebSphere MQ / IBM MQ for z/OS Migration Guide - HTML version.
- To download the guide as a PDF file, click the following link: IBM WebSphere MQ / IBM MQ for z/OS Migration Guide - PDF file.

Downloadable versions of the IBM MQ product documentation

If you prefer to use the IBM MQ Version 9.0 product documentation offline, you can download it through the links on the IBM MQ documentation library web page, either as a downloadable package or as a set of PDF files. The PDF files are also available from the links in IBM MQ Version 9.0 PDF documentation.

The version naming scheme for IBM MQ for Multiplatforms

Multi

From IBM MQ Version 9.0, releases have a three digit Version, Release, and Modification (VRM) code or a four-digit Version, Release, Maintenance, and Fix (VRMF) level code.

From IBM MQ Version 9.0, the full version of IBM MQ is described by a three-digit or four-digit number.

LTS For the Long Term Support (LTS) release model, the number consists of a four-digit VRMF code.

CD For the Continuous Delivery (CD) release model, the number consists of a three-digit VRM code on z/OS and a four digit VRMF code on Multiplatforms, where the final digit is always a zero.

The VRMF acronym stands for:

`Version.Release.Modification.Fix`

The two release types are distinguishable by the modification number in the version.release.modification (`v.r.m`) release identifier.

IBM MQ Version 9.0.0 has a zero modification number and is designated as LTSR.

CD Continuous Delivery releases have a non-zero modification number, for example, 9.0.1, 9.0.2, and so on.

The version and release parts of the code are significant; they identify the service life of a release. To run a queue manager at a different VR level, you must migrate the queue manager, its applications, and the environment in which it runs. Depending on the migration path, the migration might require more or less effort.

`7.5`, `7.0.1.1`, and `8.0.0.4` are examples of IBM MQ version codes for previous versions.

You can find the full version level of an IBM MQ installation by typing the command **`DSPMQVER`**, or **`DSPMQMVER`** on IBM i. It returns the full three-digit VRM, or four-digit VRMF code.

Versions and releases of IBM MQ are known by the first two digits of the VRMF code. The two digits are sometimes prefixed by a `V`, such as `V9.0`. A version of IBM MQ always has a release level, even if it is the first release in a version.

The first release is normally labeled `V` *`x`*`.0`, for example, IBM MQ V8.0. Occasionally, the first release of a version on a specific platform is not labeled `V` *`x`*`.0`. It is numbered to correspond the command level that has been implemented on the platform.

The third digit in the VRMF identifies the modification level of a release. A change in the third digit does not change the release. For example, after upgrading IBM MQ to modification level `8.0.1`, the release of IBM MQ remains 8.0. However the command level does change to `801`.

Notes:

1. Multi Backward migration is not possible. To be able to restore an earlier version or release level of a queue manager, you must back it up before upgrading. If you do restore it, you restore the queue manager, and its data, to the state it was in when you backed it up.
2. z/OS Backward migration is possible only if you are using the LTSR model.

A new version or release has a new service end date. New modification levels generally do not result in a new service end date. But if a modification level is announced, then a new service end date might be announced too.

The fourth digit in the VRMF code represents the fix pack level. For example, the first fix pack of the IBM MQ Version 9.0.0 LTS release is numbered 9.0.0.1. Fix levels do not affect the command level of the queue manager. No migration is required, and fix levels do not affect the service end date of a release.

Attention: From IBM MQ Version 9.0, the name is changed, for example, to 9.0.0-IBM-MQ-Windows-FP0001.

Refresh packs and fix packs for a particular version/release are cumulative, from the initial release. You can apply any higher numbered refresh, or fix pack, of the same version/release to upgrade directly to that version level. You do not have to apply the intervening fixes. Refresh packs and fix packs are obtained as service through Fix Central.

The latest modification level is also used to refresh the version of IBM MQ available through Electronic Software Download using Passport Advantage, or on physical media.

When you order IBM MQ you receive the latest version of the LTS, or CD product, depending on which support model your enterprise is using.

The result of installing a manufacturing refresh is almost the same as applying the refresh pack to an earlier fix level of IBM MQ. There is one important difference. Refresh packs are applied using a maintenance procedure, manufacturing refreshes are installed using an installation procedure. You can "unapply" a refresh pack to return to the previous fix level you had installed. You can only uninstall a manufacturing refresh, which removes IBM MQ from your system.

In addition to fixes packaged as refresh packs and fix packs, you can also obtain interim fixes for IBM MQ. You get these from Fix Central. Interim fixes are also known as emergency or test fixes, and are known collectively as interim fixes. The naming scheme for refresh and fix packs extends to interim fixes. Interim fixes are known either by their fix name, or by the list of APARs they fix.

When you apply new fix packs or refresh packs, all interim fixes are removed. The documentation with the fix pack or refresh pack tells you if the APARS associated with the interim fixes you have applied have been fixed. If they have not, check to see if there are new interim fixes, at the new level, for the APARs that concern you. If there are not, consult service. They might either tell you to reapply the interim fix, or supply a new interim fix.

Related concepts:

"The version naming scheme for IBM MQ for z/OS"
On IBM MQ for z/OS, releases have a three digit Version, Release, and Modification (VRM) code. To run a queue manager at a different VRM level, you must migrate the queue manager, its applications, and the environment in which it runs. Depending on the migration path, the migration might require more or less effort.

Related tasks:

"Maintaining and migrating IBM MQ" on page 565
Maintenance, upgrade, and migration have three distinct meanings for IBM MQ. The definitions are described here. The following sections describe the various concepts associated with migration, followed by the various tasks needed; these tasks are platform-specific where needed.

The version naming scheme for IBM MQ for z/OS

z/OS

On IBM MQ for z/OS, releases have a three digit Version, Release, and Modification (VRM) code. To run a queue manager at a different VRM level, you must migrate the queue manager, its applications, and the environment in which it runs. Depending on the migration path, the migration might require more or less effort.

The release level of IBM MQ for z/OS is described by a three-digit VRM code. This applies to both the Long Term Support (LTS) release and the Continuous Delivery (CD) release model.

The two release types are distinguishable by the modification number in the version.release.modification (`v.r.m`) release identifier.

LTS IBM MQ Version 9.0.0 has a zero modification number and is designated as LTSR.

CD Continuous Delivery releases have a non-zero modification number, for example, 9.0.1, 9.0.2, and so on.

`7.0.1`, `7.1.0`, `8.0.0`, and `9.0.0` are examples of IBM MQ for z/OS release level codes.

On z/OS, a release of IBM MQ always has a three-digit VRM code, even if the release is the first release in a version, such as `8.0.0`. IBM MQ for z/OS follows a convention of changing the VRM when the product is installed by SMP/E with a new FMID.

You can modify existing libraries, without changing the FMID, by applying PTFs. You cannot upgrade existing libraries to an FMID or release level by applying PTFs.

The release level of a z/OS queue manager is written to the operator console in the message CSQY000I.

The command level of a queue manager is a three-digit VRM code. You can look at the queue manager command level in the queue manager property panel in IBM MQ Explorer. An IBM MQ program can call MQINQ, passing the `MQIA_COMMAND_LEVEL` selector, to obtain the command level of the queue manager it is connected to.

The VRM code, or release level, is significant in two respects. Changing the release level that a queue manager runs at, requires migration of the queue manager. It also requires attention to the PTF level of other queue managers that are in the same queue-sharing group. It is also significant because each release level has its own service life, and end of service date.

The service life depends on the VRM. Each release level has its own service end date. So, for example, `8.0.0`, on z/OS, has a different service end date from `8.0.1`. See IBM Software Support lifecycle policy. Click through to WebSphere product lifecycle dates, to find the service life and end of service for different releases of IBM MQ.

Note: Backward migration z/OS is possible only if you are using the Long Term Support model.

Related concepts:

"Upgrade and migration of IBM MQ on z/OS" on page 785
You can install new releases of IBM MQ to upgrade IBM MQ to a new release, or version level. Multiple installations at the same or different levels can coexist on the same z/OS instance. Running a queue manager at a higher level requires migration.

"The version naming scheme for IBM MQ for Multiplatforms" on page 567
From IBM MQ Version 9.0, releases have a three digit Version, Release, and Modification (VRM) code or a four-digit Version, Release, Maintenance, and Fix (VRMF) level code.

Migrating queue managers: new-function and maintenance fix packs

This topic has been removed, as the information contained is not compatible with the Continuous Delivery and Long Term Support release models used in IBM MQ Version 9.0, or later.

Applying maintenance to IBM MQ

Maintenance is the application of a reversible fix. Any changes to queue manager data are compatible with the previous code level.

About this task

LTS **Long Term Support releases**

Multi On Multiplatforms, if your enterprise is using the Long Term Support (LTS) release model, maintenance is the process of applying interim fixes or fix-packs.

z/OS On z/OS, from Version 9.0, if your enterprise is using the Long Term Support (LTS) release model, you use Program Temporary Fixes (PTFs) to apply maintenance to the installed code.

CD **Continuous Delivery releases**

On all supported platforms, if your enterprise is using the Continuous Delivery (CD) release model, you can select which updates your enterprise requires as each CD release replaces the prior one for that version of IBM MQ.

For more information about Long Term Support and Continuous Delivery releases, see IBM MQ Release Types.

An important characteristic of applying maintenance is that it must be reversible. Reversibility implies two things:

1. The previous level of code is fully restored.
2. Changes that are made to IBM MQ objects are compatible. Changes are things like the creation or deletion of persistent messages, changes to queue managers, channels, topics, and queues. New and modified objects continue to work correctly with the restored level of code.

The reversibility of a maintenance package limits the extent of functional changes that are included in a maintenance package. No irreversible changes are included in a maintenance package. But, reversibility has limits. A maintenance package might include new programming and administrative interfaces. If you build new or modified applications to use the new interfaces, those applications do not work, if the maintenance package is removed.

Multi-instance queue managers are a good example. Should you remove the Version 7.0.1 fix pack that upgraded Version 7.0, then multi-instance queue manager functions no longer work. However, the queue managers continue to function correctly as single instance queue managers in Version 7.0.

On a smaller scale, a fix pack or interim fix might introduce a new configuration parameter to solve a problem. If you remove the fix pack or interim fix, although the new interface introduced by the change is not available any more, IBM MQ works with any objects that have been changed by the configuration parameter. For example, a new Java system property might introduce a parameter to set a code page for queue manager data conversion. The fix does not change any existing persistent queue manager state information. It can be removed, and the queue manager continues to work as before, but without the capability introduced in the fix.

On different platforms, you employ different mechanisms to install and maintain software releases. Installing a release at a new maintenance level, and applying maintenance level updates to updatee an earlier release to the same maintenance level, have different results.

When you update the maintenance or fix level of IBM MQ by applying a regular maintenance level update, you can reverse the update by removing the fix. When you update the maintenance or fix level of IBM MQ by applying a maintenance level update containing a new function, you can reverse that update and all previously reversible updates until a queue manager associated with the installation enables the new function.

Maintenance levels and fix levels are both supplied from the service site, Fix Central. Fix central has a function to tell you what updates you can apply to the current level of your system. If you back out a maintenance level update, it returns IBM MQ code to the same level of code as before applying the maintenance level update.

Procedure

- Multi For information on how to apply and remove fix packs on Multiplatforms, follow the appropriate link in Applying and removing maintenance for the platform that your enterprise uses.

Table 75. Applying and removing maintenance

Apply	Remove
AIX AIX	AIX
HP-UX HP-UX	HP-UX
Linux Linux	Linux
Solaris Solaris	Solaris
IBM i IBM i	IBM i
Windows Windows	Windows

- z/OS For z/OS, see "Applying and removing maintenance on z/OS" on page 617.

Related concepts:

"The version naming scheme for IBM MQ for Multiplatforms" on page 567
From IBM MQ Version 9.0, releases have a three digit Version, Release, and Modification (VRM) code or a four-digit Version, Release, Maintenance, and Fix (VRMF) level code.

"Multi-installation queue manager coexistence on UNIX, Linux, and Windows" on page 668
You can install multiple copies of IBM MQ for UNIX, Linux, and Windows on the same server. The installations must be at Version 7.1 or later, with one exception. One Version 7.0.1 installation, at fix pack level 6, or later, can coexist with multiple Version 7.1, or later installations.

"Queue manager coexistence" on page 664
Queue managers, with different names, can coexist on any server as long as they use the same IBM MQ installation. On z/OS, UNIX, Linux, and Windows, different queue managers can coexist on the same server and be associated with different installations.

Related information:

Backing up and restoring a queue manager

IBM MQ maintenance tasks on Multiplatforms - general information

Multi

When you apply and remove maintenance level updates to IBM MQ, no migration is required. Maintenance level updates are applied either as a fix pack, or by manually applying an interim fix.

About this task

This information has moved. See "Applying maintenance to IBM MQ" on page 570.

New function in maintenance level updates

Multi

This topic has been removed, as the information contained is not compatible with the Continuous Delivery and Long Term Support release models used in IBM MQ Version 9.0, or later.

Querying the maintenance level

Query the IBM MQ maintenance level by running the **dspmqver** command

About this task

In previous versions of the product, after an update to the initial installation, the version indicates the maintenance level to which the product has been updated. For example, before applying any maintenance, the version is 8.0.0.2. As maintenance is applied the last digit is updated, for example to 8.0.0.3.

From IBM MQ Version 9.0 there are two types of release; a Long Term Support (LTS) release and Continuous Delivery (CD) release. For more information, see IBM MQ Release Types.

Procedure

To view the version use the `dspmqver` command. At a command prompt, enter the following command: `dspmqver`. The resulting messages include the IBM MQ version number, which shows the maintenance level.

Related information:

dspmqver

Applying and removing maintenance on Windows

Windows

Maintenance tasks associated with IBM MQ on Windows are grouped in this section.

Procedure

- To apply maintenance level server updates, see "Applying maintenance level server updates on Windows."
- To apply maintenance level client updates, see "Applying maintenance level client updates on Windows" on page 579.
- To remove updates and revert to the previous maintenance level using the Windows installer, see "Reverting to the previous maintenance level on Windows" on page 580.
- For information on how to use multiple installations of IBM MQ on the same server to control the release of maintenance fixes, see "Staging maintenance fixes on Windows" on page 583.
- For information on how to use use multi-instance queue managers to reduce the outage caused by applying maintenance updates, see "Applying maintenance level updates to multi-instance queue managers on Windows" on page 577.

Related tasks:

"Querying the maintenance level" on page 572
Query the IBM MQ maintenance level by running the **`dspmqver`** command

Applying maintenance level server updates on Windows

Windows

You can apply maintenance level server updates to IBM MQ for Windows either interactively or silently.

Before you begin

1. If you are running on a server with multiple IBM MQ installations, you must identify the installation. Make sure that the commands you enter run against the correct installation; see setmqenv.
2. Download the maintenance package from the IBM MQ Support website.
3. If User Account Control (UAC) is enabled, the user who does the installation must have Administrative authority. You must elevate any command or command prompt by selecting **Run as Administrator**. If you do not, the error `AMQ4353` is written in the installation log.

Procedure

1. Log on as an Administrator.
2. Stop all applications using the IBM MQ installation.

   If you use the Managed File Transfer (MFT) component, ensure that any MFT agents have finished all of the file transfers that they were engaged in. There should be no incomplete transfers associated with the agents, and their SYSTEM.FTE.STATE queues should contain no messages.
3. End all the activity of queue managers associated with the IBM MQ installation.
   a. Run the **dspmq** command to list the state of all the queue managers on the system.

      Run either of the following commands from the installation that you are updating:
      ```
      dspmq -o installation -o status
      dspmq -a
      ```
      dspmq -o installation -o status displays the installation name and status of queue managers associated with all installations of IBM MQ.

      dspmq -a displays the status of active queue managers associated with the installation from which the command is run.
   b. Use the MQSC command **DISPLAY LSSTATUS** to list the status of listeners associated with a queue manager, as shown in the following example:
      ```
      echo DISPLAY LSSTATUS(*) STATUS | runmqsc QmgrName
      ```
   c. Run the **endmqm** command to stop each running queue manager associated with this installation.
      ```
                 ┌─-c─┐
      ►►─endmqm──┼─-w─┼─QmgrName──────────────────────────────────────────►◄
                 ├─-i─┤
                 └─-p─┘
      ```
      The **endmqm** command informs an application that the queue manager it is connected to is stopping; see Stopping a queue manager.

      For the maintenance to proceed, applications must respond to an **endmqm** command by disconnecting from the queue manager and releasing any IBM MQ libraries they have loaded. If they do not, you must find another way to force applications to release IBM MQ resources, such as by stopping the applications.

      You must also stop applications that are using the client libraries that are part of the installation. Client applications might be connected to a different queue manager, running a different installation of IBM MQ. The application is not informed about queue managers in the current installation being shut down.

      Any applications that continue to have IBM MQ shared libraries from the installation loaded prevent you applying IBM MQ maintenance. An application might disconnect from a queue manager, or be forcibly disconnected, but keep an IBM MQ shared library loaded.

      Note: "Applying maintenance level updates to multi-instance queue managers on Windows" on page 577 describes how to apply maintenance to a multi-instance queue manager. A multi-instance queue manager can continue to run on one server, while maintenance is applied to another server.
   d. Stop any listeners associated with the queue managers, using the command:
      ```
      endmqlsr -m QMgrName
      ```
4. Stop the IBM MQ service for the installation.
   a. Right-click the **IBM MQ** icon in the taskbar > click **Stop IBM MQ**.
5. Load and apply the maintenance files for server installations:
   - Interactively:
     a. Open the folder where the maintenance package has been extracted.
     b. Right-click on the maintenance program and select **Run as administrator** to start the loading process.

c. Choose your installation language, and click **OK**.

d. Continue to follow the instructions on screen.

   If you choose to load the files without applying them to an installation, you can apply the files later, as described in step 6 on page 576

- Silently:

  a. Open the folder where the maintenance package has been extracted.

  b. Modify the response file, `silent_install.resp`. For details on the properties you can specify in the response file, see Table 76.

Table 76. Properties used to install or uninstall a maintenance update

Property	**Value**	**Description**
`MQPLOG`	*`path\file_name`*	Pass a valid path to specify the log to be used during installation/uninstallation, for example `MQPLOG="C:\TEMP\UPDATEINSTALL.LOG"` If `MQPLOG` is not specified (which is the case if you start maintenance by clicking the **Apply fix pack n.n.n.n** icon in the IBM MQ program group) the log name used by default will be `amqicsdn.txt` in your TEMP directory (`%TEMP%`).
`MQPINSTALLATIONNAME`	*`Installation name`*	The name of the installation to maintain. If there is only one installation (of any level) on the machine, this argument can be safely omitted. If there is more than one installation on the machine, `amqicsdn.exe` checks the value of `MQPINSTALLATIONNAME`. If one is not supplied, or the one that is supplied is unsuitable, then a GUI selection box appears. This selection box provides a list of installations to which this fix pack is applicable. If none is applicable, then `amqicsdn.exe` issues error message `AMQ4781` and ends.
`MQPBACKUPPATH`	*`path`*	Specifies the directory to back up into during installation, for example `MQPBACKUPPATH="C:\BACKUP"` The directory, and any intermediate directories, you specify must already exist. If any one of the directories does not already exist, the installation fails.
`MQPREBOOT`	`0|1`	Specifies what to do when a reboot is required, for example `MQPREBOOT=1`. If no value is supplied, you are prompted for what to do next. If `MQPREBOOT` is set to 0, a reboot is suppressed If `MQPREBOOT` is set to 1, the reboots go ahead without prompting.
`MQPINUSEOK`	`0|1`	Specifies whether to continue even if a file is found to be currently locked by another application. If you choose to continue even if a file is locked by another application, then you must reboot to complete fix pack installation. If no value is supplied, or if `MQPINUSEOK` is set to `0`, the installation fails if files are found to be in use by other applications. If `MQPINUSEOK` is set to `1`, the installation is deferred until you reboot.

  c. Open an elevated command prompt in the directory where the maintenance program was extracted.

   d. Start the silent loading by entering the following command:
      ```
      executableName -f responseFile
      ```
      where:
      - *executableName* is the name of the maintenance package. For example, for Version 9.0.0, Fix Pack 1: `9.0.0-IBM-MQ-Windows-FP0001.exe`.
      - *responseFile* is the full path and name of the response file.
6. Optional: Apply the maintenance to other server installations on the system:
   - Interactively:
     a. From the Windows start menu, select **Start > Programs > IBM MQ > Apply Fix Pack *V.R.M.L***.
        where
          V is the version number
          R is the release number
          M is the modification number
          L is the level of modification
     b. Continue to follow the instructions on screen.
   - Silently:
     a. Open an elevated command prompt and navigate to the directory where the maintenance program was loaded. By default, the path is `C:\Program Files (x86)\IBM\source\WebSphere MQ` *`V.R.M.L`*
        where
          V is the version number
          R is the release number
          M is the modification number
          L is the level of modification
     b. Enter the following command:
        ```
        amqicsdn MQPINSTALLATIONNAME= name MQPSILENT=1
        ```
        where *name* is the name of the installation that you want to apply maintenance to.
        You can add other properties to the command, as listed in Table 76 on page 575.
7. Optional: Uninstall the fix pack files from your machine. After installing the fix pack files and applying the maintenance to all the server installations that you want to update, you can either uninstall the fix pack files from your machine or leave them installed for future use.
   To uninstall the fix pack files use the **Control Panel... Programs and Features** panel, select the item **IBM MQ (fix pack 9.0.0.x) files**, and click **Uninstall**.
   - Uninstalling these files does NOT remove them from the installations to which you have already applied the maintenance. If that is what you intend, you should instead follow the instructions in "Reverting to the previous maintenance level on Windows" on page 580.
   - If you add any installable features at a later time, you must reapply the maintenance to update the added feature(s).

   Note: The fix pack files contain a JRE, so if you chose not to install a JRE in your IBM MQ installation for local policy reasons, you may want to uninstall the fix pack files as soon as you have finished applying the update to your installation(s).

What to do next

On a server installation, you must restart the IBM MQ taskbar application manually after the maintenance application completes.

The IBM MQ service is restarted automatically on the server, but the taskbar application is not restarted for any logged in sessions. Start the taskbar application in one of three ways:

1. Start the taskbar application manually from the start menu.
2. Log off and log back on again.
3. Run the command:
   ```
   MQ_INSTALLATION_PATH\bin\amqmtbrn.exe -Startup
   ```

Related information:

dspmq (display queue managers)

DISPLAY LSSTATUS

Stopping a queue manager

endmqm (end queue manager)

endmqlsr (end listener)

Applying maintenance level updates to multi-instance queue managers on Windows

Applying maintenance level updates to multi-instance queue managers on Windows

Windows

On Windows platforms, you can use multi-instance queue managers to reduce the outage caused by applying maintenance updates.

Before you begin

Before starting this task, read through the prerequisites described in *Before you begin* in "Applying maintenance level server updates on Windows" on page 573

Before starting this task, see the Maintenance is applied to the IBM MQ installation on a server and not to individual queue managers. Before you apply maintenance, you must stop all the queue managers, and any IBM MQ service, on a server.

If you want a queue manager to keep running while maintenance is applied, you must configure it as a multi-instance queue manager, and have a standby instance running on another server. If the queue manager that you want to keep running is an existing single instance queue manager, you must convert it to a multi-instance queue manager. For prerequisites and guidance how to create a multi-instance queue manager, see Multi-instance queue managers.

You can create a multi-instance queue manager from Version 7.0.1 onwards. If you are running multi-instance queue managers, you then can apply a maintenance update to a running queue manager by switching the active instance to a different server.

Typically, active and standby installations are maintained at the same maintenance level. Consult the maintenance instructions for each update. Consult the instructions to see if it is possible to run the active and standby instances at different maintenance levels. Check whether fail over from higher to lower, or only lower to higher maintenance level is possible.

The instructions for applying a maintenance update might require you to stop a multi-instance queue manager completely.

If you have a primary server for running active queue manager instances, and a secondary server that runs standby instances, you have a choice of updating the primary or secondary server first. If you update the secondary server first, you must switch back to the primary server when both servers have been updated.

If you have active and standby instances on several servers, you must plan in what order you update the servers to minimize the disruption caused by ending the active instances on each server you update.

About this task

Follow these steps to apply maintenance to a multi-instance queue manager on Windows.

Procedure

1. Log on as an Administrator.
2. Stop all applications using the IBM MQ installation.

   If you use the Managed File Transfer (MFT) component, ensure that any MFT agents have finished all of the file transfers that they were engaged in. There should be no incomplete transfers associated with the agents, and their SYSTEM.FTE.STATE queues should contain no messages.
3. Find out the current state of the queue managers and their associated listeners associated with the IBM MQ installation.
   a. From the installation that you are updating, use the **dspmq** command to list the state of the queue managers:
      - To display the installation name and status of queue managers associated with all installations of IBM MQ, run the following command:
        ```
        dspmq -o installation -o status
        ```
      - To display the status of active queue managers associated with the installation from which you are running the command, run the following command:
        ```
        dspmq -a
        ```
   b. Use the MQSC command **DISPLAY LSSTATUS** to list the status of listeners associated with a queue manager, as shown in the following example:
      ```
      echo "DISPLAY LSSTATUS(*) STATUS" | runmqsc QmgrName
      ```
4. Use the **endmqm** command to stop each running queue manager associated with this installation.
   - If the queue manager is running as standby, run the **endmqm** command to end the standby as shown in the following example:
     ```
     endmqm -x QMgrName
     ```
   - If the queue manager is running as the active instance, run the **endmqm** command to end the active instance and transfer control to the standby instance as shown in the following example:
     ```
     endmqm -shutdown_option -s QMgrName
     ```

     where *-shutdown_option* is an optional parameter specifying the type of shutdown. For more information about optional parameters for the **endmqm** command, see endmqm.

     If there is no standby instance running, and the command fails, start a standby instance on a different server.
   - If a queue manager is running as a single instance queue manager, stop the queue manager. In the case of a single queue manager you have no alternative but to stop the queue manager before applying the maintenance update. For more information about how to stop a queue manager, see Stopping a queue manager.

   Stop any listeners associated with the queue managers by using the **endmqlsr** command as shown in the following example:
   ```
   endmqlsr -m QMgrName
   ```

   After you complete this step, no queue manager instances are left running on the server that you intend to update.
5. Apply maintenance to the IBM MQ server. Follow the instructions in “Applying maintenance level server updates on Windows” on page 573, starting from Step 4.

6. When you have completed the maintenance update, use the **strmqm** command to restart all the queue managers on the IBM MQ server, permitting standby instances, as shown in the following example:
   ```
   strmqm -x QmgrName
   ```
7. Repeat the procedure on the standby server, to update its maintenance level.
8. If necessary, switch the active instances back to the primary servers:
   a. Stop the instances by using the **endmqm** command as shown in the following example:
      ```
      endmqm -shutdown_option -s QMgrName
      ```
   b. Restart the instances by using the **strmqm** command as shown in the following example:
      ```
      strmqm -x QmgrName
      ```

Related information:

dspmq (display queue managers)

DISPLAY LSSTATUS

Stopping a queue manager

endmqm (end queue manager)

endmqlsr (end listener)

strmqm (start queue manager)

Applying maintenance level client updates on Windows

Windows

You can apply maintenance level updates to IBM MQ for Windows clients either interactively or by performing a silent MSI update.

About this task

You can apply and remove maintenance from an IBM MQ client interactively or by using the **msiexec** command to perform a silent MSI update.

Interactive client update

On the client installation media, navigate to the `\Windows\MSI\` directory, then run the `Setup.exe` file.

Silent client update

As an alternative method for applying maintenance to IBM MQ client systems, you can use the command **msiexec** from the command line to perform a silent MSI update.

To update a computer with only a single installation, you can use a command similar to the following example:

```
msiexec /i "PATH\Windows\MSI\IBM MQ.msi" /l*v install_log_path
/q TRANSFORMS="1033.mst" REINSTALL=ALL REINSTALLMODE=vomus
```

For a multi installation computer with multiple clients, you can update a single client by using a command similar to the following example:

```
msiexec /i "PATH\Windows\MSI\IBM MQ.msi" /l*v install_log_path
/q TRANSFORMS=":InstanceId2.mst;1033.mst" REINSTALL=ALL REINSTALLMODE=vomus
```

Results

When the maintenance completes you can query the maintenance level by running the **dspmqver** command. For more details, see "Querying the maintenance level" on page 572.

Reverting to the previous maintenance level on Windows

Windows

You can remove updates and revert to the previous maintenance level of IBM MQ by using the Windows installer.

Before you begin

1. If you are running on a server with multiple IBM MQ installations, you must identify the installation. Make sure that the commands you enter run against the correct installation; see setmqenv.
2. If User Account Control (UAC) is enabled, the user who does the installation must have Administrative authority. You must elevate any command or command prompt by selecting **Run as Administrator**. If you do not, the error `AMQ4353` is written in the installation log.

About this task

If you applied maintenance to IBM MQ, you can restore IBM MQ to a previous level of maintenance.

If you installed IBM MQ at a particular maintenance level, a *Manufacturing Refresh*, you cannot restore IBM MQ to an earlier maintenance level.

Procedure

1. Log on as an Administrator.
2. Stop all applications using the IBM MQ installation.

   If you use the Managed File Transfer (MFT) component, ensure that any MFT agents have finished all of the file transfers that they were engaged in. There should be no incomplete transfers associated with the agents, and their SYSTEM.FTE.STATE queues should contain no messages.
3. End all the activity of queue managers associated with the IBM MQ installation.
   a. Run the **dspmq** command to list the state of all the queue managers on the system.

      Run either of the following commands from the installation that you are updating:

      ```
      dspmq -o installation -o status
      dspmq -a
      ```

      dspmq -o installation -o status displays the installation name and status of queue managers associated with all installations of IBM MQ.

      dspmq -a displays the status of active queue managers associated with the installation from which the command is run.
   b. Use the MQSC command **DISPLAY LSSTATUS** to list the status of listeners associated with a queue manager, as shown in the following example:

      ```
      echo DISPLAY LSSTATUS(*) STATUS | runmqsc QmgrName
      ```
   c. Run the **endmqm** command to stop each running queue manager associated with this installation.

      ```
                    ┌-c┐
      ►►─endmqm─────┼-w┼──QmgrName──────────────────────────────────►◄
                    ├-i┤
                    └-p┘
      ```

      The **endmqm** command informs an application that the queue manager it is connected to is stopping; see Stopping a queue manager.

      For the maintenance to proceed, applications must respond to an **endmqm** command by disconnecting from the queue manager and releasing any IBM MQ libraries they have loaded. If they do not, you must find another way to force applications to release IBM MQ resources, such as by stopping the applications.

You must also stop applications that are using the client libraries that are part of the installation. Client applications might be connected to a different queue manager, running a different installation of IBM MQ. The application is not informed about queue managers in the current installation being shut down.

Any applications that continue to have IBM MQ shared libraries from the installation loaded prevent you applying IBM MQ maintenance. An application might disconnect from a queue manager, or be forcibly disconnected, but keep an IBM MQ shared library loaded.

Note: "Applying maintenance level updates to multi-instance queue managers on UNIX and Linux" on page 611 describes how to apply maintenance to a multi-instance queue manager. A multi-instance queue manager can continue to run on one server, while maintenance is applied to another server.

d. Stop any listeners associated with the queue managers, using the command:

```
endmqlsr -m QMgrName
```

4. Stop the IBM MQ service for the installation.
   a. Right-click the **IBM MQ** icon in the taskbar > click **Stop IBM MQ**.
5. Remove the maintenance interactively, or silently using a command.
   - Interactively:
     a. For each installation of IBM MQ that has had maintenance applied, you are presented with one of the following icons in the Windows start menu:
        1) **Start > Programs > IBM MQ > Remove Refresh Pack *V.R.M.L* (*installation_name*)**
        2) **Start > Programs > IBM MQ > Remove Fix Pack *V.R.M.L* (*installation name*)**

        where

        V is the version number

        R is the release number

        M is the modification number

        L is the level of modification
     b. Select the installation you want to maintain and click **Remove** to start the process.

     This returns the installation to the state it was in before the maintenance package was applied.
   - Silently:
     a. Open an elevated command prompt and enter the following command:

        ```
        amqicsdn.exe MQPINSTALLATIONNAME= name MQPUNINST=1 MQPSILENT=1
        ```

        where *name* is the name of the installation that you want to remove maintenance from.

        You can add other properties to the command, as listed in Table 77.

Table 77. Properties used to install or uninstall a maintenance update

Property	**Value**	**Description**
MQPLOG	*path\file_name*	Pass a valid path to specify the log to be used during installation/uninstallation, for example MQPLOG="C:\TEMP\UPDATEINSTALL.LOG" If MQPLOG is not specified (which is the case if you start maintenance by clicking the **Apply fix pack n.n.n.n** icon in the IBM MQ program group) the log name used by default will be amqicsdn.txt in your TEMP directory (%TEMP%).

Table 77. Properties used to install or uninstall a maintenance update (continued)

Property	Value	Description
MQPINSTALLATIONNAME	*Installation name*	The name of the installation to maintain. If there is only one installation (of any level) on the machine, this argument can be safely omitted. If there is more than one installation on the machine, amqicsdn.exe checks the value of MQPINSTALLATIONNAME. If one is not supplied, or the one that is supplied is unsuitable, then a GUI selection box appears. This selection box provides a list of installations to which this fix pack is applicable. If none is applicable, then amqicsdn.exe issues error message AMQ4781 and ends.
MQPBACKUPPATH	*path*	Specifies the directory to back up into during installation, for example MQPBACKUPPATH="C:\BACKUP" The directory, and any intermediate directories, you specify must already exist. If any one of the directories does not already exist, the installation fails.
MQPREBOOT	0|1	Specifies what to do when a reboot is required, for example MQPREBOOT=1. If no value is supplied, you are prompted for what to do next. If MQPREBOOT is set to 0, a reboot is suppressed If MQPREBOOT is set to 1, the reboots go ahead without prompting.
MQPINUSEOK	0|1	Specifies whether to continue even if a file is found to be currently locked by another application. If you choose to continue even if a file is locked by another application, then you must reboot to complete fix pack installation. If no value is supplied, or if MQPINUSEOK is set to 0, the installation fails if files are found to be in use by other applications. If MQPINUSEOK is set to 1, the installation is deferred until you reboot.

6. Optional: If you no longer need the maintenance files that were loaded onto the system before maintenance was applied, you can remove them using **Add/Remove programs** or **Programs and Features** from the Control Panel. If you want to remove a maintenance file silently, run the following command:

```
patch_install_files\_IBM MQ (fix pack V.R.M.L files)_installation\Change IBM MQ
(fix pack V.R.M.L files) Installation.exe" -i silent
```

   where *patch_install_files* is the installation directory where maintenance files are installed.

   By default, this directory is C:\Program Files (x86)\IBM\source\WebSphere MQ *V.R.M.L*

   Notes:

   a. Run the command from outside the directory, otherwise the directory is not removed.
   b. If you omit **-i silent**, the command initiates the Graphical User Interface installer.

What to do next

On a server installation, you must restart the IBM MQ taskbar application manually after the maintenance application completes.

The IBM MQ service is restarted automatically on the server, but the taskbar application is not restarted for any logged in sessions. Start the taskbar application in one of three ways:

1. Start the taskbar application manually from the start menu.
2. Log off and log back on again.
3. Run the command:

```
MQ_INSTALLATION_PATH\bin\amqmtbrn.exe -Startup
```

Related information:
dspmq
Stopping a queue manager
DISPLAY LSSTATUS
endmqm (end queue manager)
endmqlsr (end listener)
Applying maintenance level updates to multi-instance queue managers on UNIX and Linux

Staging maintenance fixes on Windows

Windows

On Windows systems, you can use multiple installations of IBM MQ on the same server to control the release of maintenance fixes.

Before you begin

Set up your configuration modeled on the first row of Figure 59 on page 584. You can apply this scenario to any version of IBM MQ from IBM WebSphere MQ Version 7.1 onwards. In this scenario it is assumed you have a number of applications and two queue managers, `QM1` and `QM2`, running on a server. IBM WebSphere MQ Version 7.0.1 is not installed on the server.

1. Install two copies of IBM MQ. In the example, they are named `Inst_1` and `Inst_2`.
2. Make `Inst_1` primary by running **`setmqinst`**.
3. Associate all the queue managers on the server with `Inst_1` by running **`setmqm`**.
4. Start all the queue managers on the server.
5. Show and connect all direct connections with the queue managers associated with `Inst_1` in IBM MQ Explorer.
6. Set up remote connections to all the queue managers in each instance of IBM MQ Explorer.

About this task

You can install multiple copies of IBM MQ on a server to stage the release of IBM MQ fixes. Figure 59 on page 584 illustrates a way of using two installations to roll out fixes. In this approach, you maintain two fix levels on a server, with the aim of getting all queue managers and applications to the production fix level before replacing the previous level on fix pack with the next level.

Which installation an application uses is driven by the queue manager an application connects to. The **`setmqm`** command associates a queue manager with an installation. You can associate a queue manager with a different installation as long as the installation is at the same or higher command level. In this example, all the installations are at the same command level. You can associate or reassociate a queue manager with either of the installations running any of the fix packs.

In the example, an application links to the primary installation. When it connects to a queue manager, IBM MQ switches the linkage to the installation associated with the queue manager; see “Multi-installation queue manager coexistence on UNIX, Linux, and Windows” on page 668.

For applications built with the link options described in the product documentation, the simplest way to configure the link library search path for IBM MQ applications is to make an installation primary. Only if it is important to pick up a fix in the IBM MQ link library itself, must you review the search path. Either you must make the installation with the IBM MQ link library fix primary, or make a local adjustment for the application, perhaps by running the **setmqenv** command.

Running commands is a different matter. Commands are always run from the primary installation, or the installation you have selected by running the **setmqenv** command. If you run a command from the wrong installation, the command fails. For example, if `QM1` is associated with `Inst_1`, running the Windows command, `Inst_2_Installation_path/bin/strmqm QM1` fails.

If you are using IBM MQ Explorer and you have two installations, you also have two IBM MQ Explorer instances. One linked to one installation, and one to the other. Each IBM MQ Explorer shows locally connected queue managers that are associated with the same installation as the instance of IBM MQ Explorer. To monitor all the queue managers on a server, set up remote connections to the queue managers associated with the other installations.

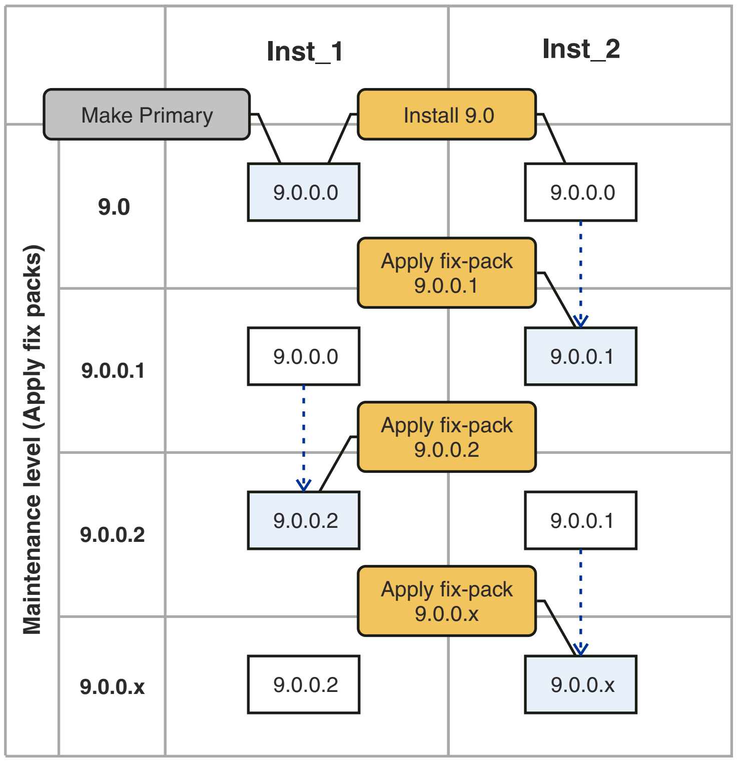


Figure 59. Rolling fix releases

Procedure

1. Download the first fix pack, for example, 7.1.0.1, when it is released.

   See Fix Central.
2. Apply the fix pack you downloaded to `Inst_2`. See "Applying and removing maintenance on Windows" on page 573.
3. Verify `Inst_2`.
4. Transfer the queue managers to `Inst_2` one at a time.
   a. Stop `QM1` and the applications connected to it.

The **endmqm** command informs an application that the queue manager it is connected to is stopping; see Stopping a queue manager.

Note: The topic, "Applying maintenance level updates to multi-instance queue managers on Windows" on page 577, describes how to apply maintenance to a multi-instance queue manager. A multi-instance queue manager can continue to run on one server, while maintenance is applied to another server.

b. Set up the local environment to the installation `Inst_2`.

```
"Inst_2_INSTALLATION_PATH\bin\setmqenv" -s
```

The `-s` option sets up the environment for the installation that runs the **setmqenv** command.

c. Associate the queue manager with `Inst_2`.

```
setmqm -m QM1 -n Inst_2
```

d. Start `QM1`

```
strmqm QM1
```

e. Repeat substeps c and d for `QM2`.

f. Set up IBM MQ Explorer for `Inst_2`.

1) Start the `Inst_2` instance of IBM MQ Explorer

Tip: On Windows, hover over the IBM MQ icon in the system tray. The hover help shows the installation name associated with the IBM MQ Explorer instance.

2) Click **IBM MQ** > **Queue Managers** > **Show/Hide Queue Managers...** >

3) Click each directly connected queue manager listed in the Hidden Queue Managers list > **Show**.

4) Click **Close**.

5. Set `Inst_2` primary.

```
"Inst_2_INSTALLATION_PATH\bin\setmqinst" -i -n Inst_2
```

6. Download the next fix pack for the version of your product, for example, 7.1.0.2, when it is released.
   See Fix Central.
7. Apply the fix pack that you have just downloaded to `Inst_1`. See "Applying and removing maintenance on Windows" on page 573.
8. Verify `Inst_1`.
9. Transfer queue managers to `Inst_1` one at a time.
   a. Follow the procedure in step 4 on page 584
   Replacing `Inst_2` by `Inst_1` in the instructions.
10. Set `Inst_1` primary.

```
"Inst_1_INSTALLATION_PATH\bin\setmqinst" -i -n Inst_1
```

11. Repeat steps 1 on page 584 to 5 on page 585 for the odd-numbered fix packs of your product.
12. Repeat steps 6 on page 585 to 10 on page 585 for the even-numbered fix packs of your product.

Related concepts:

"Queue manager coexistence" on page 664
Queue managers, with different names, can coexist on any server as long as they use the same IBM MQ installation. On z/OS, UNIX, Linux, and Windows, different queue managers can coexist on the same server and be associated with different installations.

"Multi-installation queue manager coexistence on UNIX, Linux, and Windows" on page 668
You can install multiple copies of IBM MQ for UNIX, Linux, and Windows on the same server. The installations must be at Version 7.1 or later, with one exception. One Version 7.0.1 installation, at fix pack level 6, or later, can coexist with multiple Version 7.1, or later installations.

Related tasks:

"Migrating IBM MQ library loading from an earlier version of the product to the latest version" on page 684
No change in the way IBM MQ libraries are loaded is normally required if you upgrade from an earlier version of the product to the latest version. You must have followed the instructions on building IBM MQ applications in the earlier version, and you must replace the earlier version with the latest version of the product. If you choose to take advantage of multi-installation in the latest version of the product, based on the side-by-side or multi-stage migration scenarios, you must modify the environment for the operating system to resolve IBM MQ dependencies for an application. Typically, you can modify the runtime environment, rather than relink the application.

Related information:

Installing IBM MQ server on Windows

Associating a queue manager with an installation

Changing the primary installation

setmqenv

setmqinst

setmqm

Applying and removing maintenance on UNIX and Linux

UNIX Linux

Maintenance tasks associated with UNIX and Linux platforms are grouped in this section.

Related tasks:
"Querying the maintenance level" on page 572
Query the IBM MQ maintenance level by running the **dspmqver** command

Applying maintenance level updates on AIX

AIX

You apply maintenance level updates to IBM MQ for AIX by using **installp**.

Before you begin

1. Ensure that you have enough disk space to apply maintenance level updates. A maintenance level update requires hard disk space for installation. In addition, the installation process might require a similar amount of disk space to save the previous level. For example, a 16 MB update might require 32 MB of space. The additional space allows a maintenance level update to be removed, and the previous level to be restored automatically.
2. If you are running on a server with multiple IBM MQ installations, you must identify the installation. Make sure that the commands you enter run against the correct installation; see setmqenv.

About this task

Stop applications using the installation and use the **installp** command, to install maintenance level updates to clients and servers. Alternatively, if the installation is in the default installation location, you can use the *System Management Interface Tool*, SMIT.

Important: You cannot go back from a later version of the product to a prior version of the product, for example from IBM MQ Version 8.0 to IBM WebSphere MQ Version 7.x.

You can apply and remove maintenance from a IBM MQ MQI client that is not installed on the same server as a queue manager. You do not have to stop any queue managers or logon as administrator. Because you do not have to stop any queue managers, do not do steps 1 to 3 in the following maintenance procedure.

Major full versions of the base product are COMMITTED by default. Fix packs on a full base version can be in APPLIED state, and you can go back one release level.

If you need the ability to revert to an earlier version, you should perform a side-by-side migration, and migrate your queue managers to the later version at any time. See "Migrating on UNIX and Linux: side-by side" on page 727 for further information.

However, if you start a queue manager under IBM MQ Version 8.0, that queue manager is automatically migrated, and cannot be downgraded to the previous version.

You require a backup of the IBM WebSphere MQ Version 7.x queue manager and data logs to restore if needed.

Procedure

1. Log in as a user in `group mqm`.
2. Stop all applications using the IBM MQ installation.

   If you use the Managed File Transfer (MFT) component, ensure that any MFT agents have finished all of the file transfers that they were engaged in. There should be no incomplete transfers associated with the agents, and their SYSTEM.FTE.STATE queues should contain no messages.
3. End all the activity of queue managers associated with the IBM MQ installation.
   a. Run the **dspmq** command to list the state of all the queue managers on the system.

Run either of the following commands from the installation that you are updating:

```
dspmq -o installation -o status
dspmq -a
```

dspmq -o installation -o status displays the installation name and status of queue managers associated with all installations of IBM MQ.

dspmq -a displays the status of active queue managers associated with the installation from which the command is run.

b. Use the MQSC command **DISPLAY LSSTATUS** to list the status of listeners associated with a queue manager, as shown in the following example:

```
echo "DISPLAY LSSTATUS(*) STATUS" | runmqsc QmgrName
```

c. Run the **endmqm** command to stop each running queue manager associated with this installation.

```
           ┌─-c─┐
►►─endmqm──┼─-w─┼─QmgrName──────────────────────────────────────────────►◄
           ├─-i─┤
           └─-p─┘
```

The **endmqm** command informs an application that the queue manager it is connected to is stopping; see Stopping a queue manager.

For the maintenance to proceed, applications must respond to an **endmqm** command by disconnecting from the queue manager and releasing any IBM MQ libraries they have loaded. If they do not, you must find another way to force applications to release IBM MQ resources, such as by stopping the applications.

You must also stop applications that are using the client libraries that are part of the installation. Client applications might be connected to a different queue manager, running a different installation of IBM MQ. The application is not informed about queue managers in the current installation being shut down.

Any applications that continue to have IBM MQ shared libraries from the installation loaded prevent you applying IBM MQ maintenance. An application might disconnect from a queue manager, or be forcibly disconnected, but keep an IBM MQ shared library loaded.

Note: "Applying maintenance level updates to multi-instance queue managers on UNIX and Linux" on page 611 describes how to apply maintenance to a multi-instance queue manager. A multi-instance queue manager can continue to run on one server, while maintenance is applied to another server.

d. Stop any listeners associated with the queue managers, using the command:

```
endmqlsr -m QMgrName
```

4. Log in as root, or switch to the superuser using the **su** command.
5. Install the update in one of the following ways:
   - Update the whole installation in the default location:
     ```
     installp -agXYd . all
     ```
   - Update selected filesets in the default location:
     ```
     installp -agXYd . list of file sets
     ```
   - Update the whole product in a non-default location using the -R flag:
     ```
     installp -R USIL_Directory -agXYd . all
     ```
   - Update selected filesets in a non-default location using the -R flag:
     ```
     installp -R USIL_Directory -agXYd . list of file sets
     ```

   USIL_Directory is the installation parent directory. IBM MQ is installed underneath the directory. For example, if /USIL1 is specified, the IBM MQ product files are located in /USIL1/usr/mqm. /USIL1/usr/mqm is known as the *MQ_INSTALLATION_PATH*.

Related information:

dspmq

Stopping a queue manager

Reverting to the previous maintenance level on AIX

AIX

You can revert to a previous maintenance level by using the *System Management Interface Tool* (SMIT).

Before you begin

1. If you are running on a server with multiple IBM MQ installations, you must identify the installation. Make sure that the commands you enter run against the correct installation; see setmqenv.

About this task

You can back out maintenance updates (fix packs) and restore your system to the previous maintenance or installation level, for any component of IBM MQ for AIX that is in the **APPLIED** state.

You can apply and remove maintenance from a IBM MQ MQI client that is not installed on the same server as a queue manager. You do not have to stop any queue managers or logon as administrator. Because you do not have to stop any queue managers, do not do steps 1 to 3 in the following maintenance procedure.

Use the following command to display the current state of the IBM MQ for AIX filesets:

```
lslpp [ -R usil ] -l "mqm*"
```

To back out a maintenance update, as the user root, issue the command:

```
installp [ -R usil ] -r "mqm*"
```

Otherwise:

Procedure

1. Log in as a user in `group mqm`.
2. Stop all applications using the IBM MQ installation.

   If you use the Managed File Transfer (MFT) component, ensure that any MFT agents have finished all of the file transfers that they were engaged in. There should be no incomplete transfers associated with the agents, and their SYSTEM.FTE.STATE queues should contain no messages.
3. End all the activity of queue managers associated with the IBM MQ installation.
   a. Run the **`dspmq`** command to list the state of all the queue managers on the system.

      Run either of the following commands from the installation that you are updating:

      ```
      dspmq -o installation -o status
      dspmq -a
      ```

      `dspmq -o installation -o status` displays the installation name and status of queue managers associated with all installations of IBM MQ.

      `dspmq -a` displays the status of active queue managers associated with the installation from which the command is run.
   b. Use the MQSC command **`DISPLAY LSSTATUS`** to list the status of listeners associated with a queue manager, as shown in the following example:

      ```
      echo "DISPLAY LSSTATUS(*) STATUS" | runmqsc QmgrName
      ```
   c. Run the **`endmqm`** command to stop each running queue manager associated with this installation.

```
            ┌─-c─┐
►►──endmqm──┼─-w─┼──QmgrName──────────────────────────────────────────►◄
            ├─-i─┤
            └─-p─┘
```

The **endmqm** command informs an application that the queue manager it is connected to is stopping; see Stopping a queue manager.

For the maintenance to proceed, applications must respond to an **endmqm** command by disconnecting from the queue manager and releasing any IBM MQ libraries they have loaded. If they do not, you must find another way to force applications to release IBM MQ resources, such as by stopping the applications.

You must also stop applications that are using the client libraries that are part of the installation. Client applications might be connected to a different queue manager, running a different installation of IBM MQ. The application is not informed about queue managers in the current installation being shut down.

Any applications that continue to have IBM MQ shared libraries from the installation loaded prevent you applying IBM MQ maintenance. An application might disconnect from a queue manager, or be forcibly disconnected, but keep an IBM MQ shared library loaded.

Note: "Applying maintenance level updates to multi-instance queue managers on UNIX and Linux" on page 611 describes how to apply maintenance to a multi-instance queue manager. A multi-instance queue manager can continue to run on one server, while maintenance is applied to another server.

d. Stop any listeners associated with the queue managers, using the command:

```
endmqlsr -m QMgrName
```

4. Log in as root, or switch to the superuser using the **su** command.
5. Open the appropriate **smit** panel using this sequence:

```
Software Installation and Maintenance
Software Maintenance and Utilities
Reject Applied Software Updates (Use Previous Version)
```

   Alternatively, use a fast path command, `smit[ty] install_update`.
6. Complete the **SOFTWARE** name field.

   Enter `mqm*` to restore all applicable file set updates to your installation.

   Note: If an option to restore only selected file set updates for IBM MQ for AIX appears, avoid it. The option results in all applicable file set updates for the maintenance update being restored.
7. Click **Enter** to reject the current maintenance level and reinstate the previous maintenance or installation level.

   a. Accept displayed default values for all other fields

   b. Dismiss the confirmation message

   The reject process starts. While the command runs, it displays progress messages terminating with an **Install Summary** table.

   a. Check the table to see which components of IBM MQ for AIX have been rejected

Related information:
dspmq
Stopping a queue manager
DISPLAY LSSTATUS
endmqm (end queue manager)
endmqlsr (end listener)
Applying maintenance level updates to multi-instance queue managers on UNIX and Linux

Applying maintenance level updates on HP-UX

HP-UX

You can apply maintenance level updates to IBM MQ for HP-UX by using **`swinstall`**.

Before you begin

1. Ensure that you have enough disk space to apply maintenance level updates. A maintenance level update requires hard disk space for installation. In addition, the installation process might require a similar amount of disk space to save the previous level. For example, a 16 MB update might require 32 MB of space. The additional space allows a maintenance level update to be removed, and the previous level to be restored automatically.
2. If you are running on a server with multiple IBM MQ installations, you must identify the installation. Make sure that the commands you enter run against the correct installation; see setmqenv.

You can apply and remove maintenance from a IBM MQ MQI client that is not installed on the same server as a queue manager. You do not have to stop any queue managers or logon as administrator. Because you do not have to stop any queue managers, do not do steps 1 to 3 in the following maintenance procedure.

About this task

1. If you want to install both the base package and the maintenance update packages, install the base package separately first. Then install the maintenance update packages.
2. Turn off the autoselect dependencies feature:
   - If you are using the interactive installer, click **Options> Change Options**. Then clear the **autoselect dependencies when marking software** check box before selecting the maintenance update package for installation.
   - If you are using the command line, type the following command:
     ```
     swinstall -x autoselect_dependencies=false
     ```

   If you are installing a fix pack from a depot that also contains the base installation image, turn off the autoselect dependencies feature before starting the installation, otherwise the attempt will fail with an error message as shown in the following example:
   ```
   Could not apply the software selection "MQSERIES,r=9.0.0.2,a=HP-UX_B.11_IA,v=IBM" because a different variant of a  d
           for this product has already been selected
   ```
3. Error messages might be seen when running **`swinstall`**, even when successfully updating an installation.

   There are two approaches you can take to handling errors in the application of maintenance.
   a. Aim for an error-free update by applying maintenance only to those components that are installed.
   b. Apply the whole maintenance package and check the error logs, error by error, ignoring the insignificant errors.

   Both approaches are described.

Many of the insignificant errors are caused by **swinstall** trying to apply updates to components that are not installed. Consider whether there are any significant errors reported with the insignificant ones.

- The following errors might not indicate a serious problem. They are written to the console, or to the **swinstall** panel.

  ```
  ERROR:  "hpux11.mycompany.com:/":
  The software dependencies for 15 products or filesets cannot be resolved.
  ```

  ```
  ERROR:  "hpux11.mycompany.com:/":
  17 filesets were determined to be skipped in the analysis phase.
  The execution phase failed for "hpux11.mycompany.com:/".
  Analysis and Execution had errors.
  ```

- The following errors might not indicate a serious problem. They are written to the `swjob` output for a **swinstall** session.

  ```
  ERROR:   17 of 20 filesets had Errors.
  3 of 20 filesets had no Errors or Warnings.
  ```

  ```
  ERROR:   The Execution Phase had errors.
  See the above output for details.
  ```

Procedure

1. Log in as a user in `group mqm`.
2. Stop all applications using the IBM MQ installation.

   If you use the Managed File Transfer (MFT) component, ensure that any MFT agents have finished all of the file transfers that they were engaged in. There should be no incomplete transfers associated with the agents, and their SYSTEM.FTE.STATE queues should contain no messages.
3. End all the activity of queue managers associated with the IBM MQ installation.
   a. Run the **dspmq** command to list the state of all the queue managers on the system.

      Run either of the following commands from the installation that you are updating:

      ```
      dspmq -o installation -o status
      dspmq -a
      ```

      dspmq -o installation -o status displays the installation name and status of queue managers associated with all installations of IBM MQ.

      dspmq -a displays the status of active queue managers associated with the installation from which the command is run.
   b. Use the MQSC command **DISPLAY LSSTATUS** to list the status of listeners associated with a queue manager, as shown in the following example:

      ```
      echo "DISPLAY LSSTATUS(*) STATUS" | runmqsc QmgrName
      ```
   c. Run the **endmqm** command to stop each running queue manager associated with this installation.

      ```
                 ┌-c┐
      ▶▶─endmqm──┼-w┼─QmgrName──────────────────────────────────────────▶◀
                 ├-i┤
                 └-p┘
      ```

      The **endmqm** command informs an application that the queue manager it is connected to is stopping; see Stopping a queue manager.

      For the maintenance to proceed, applications must respond to an **endmqm** command by disconnecting from the queue manager and releasing any IBM MQ libraries they have loaded. If they do not, you must find another way to force applications to release IBM MQ resources, such as by stopping the applications.

      You must also stop applications that are using the client libraries that are part of the installation. Client applications might be connected to a different queue manager, running a different installation of IBM MQ. The application is not informed about queue managers in the current installation being shut down.

Any applications that continue to have IBM MQ shared libraries from the installation loaded prevent you applying IBM MQ maintenance. An application might disconnect from a queue manager, or be forcibly disconnected, but keep an IBM MQ shared library loaded.

Note: "Applying maintenance level updates to multi-instance queue managers on UNIX and Linux" on page 611 describes how to apply maintenance to a multi-instance queue manager. A multi-instance queue manager can continue to run on one server, while maintenance is applied to another server.

d. Stop any listeners associated with the queue managers, using the command:

```
endmqlsr -m QMgrName
```

4. Log in as root, or switch to the superuser using the **su** command.
5. Make your current directory the location of the *Service_update_package*.

   The file name of the *Service_update_package* follows the pattern `hp-Uxxxx.v11`. You must prefix *Service_update_package* with the absolute path to the installation file. To save typing, construct the path using the `$PWD` variable.
6. Run the HP-UX command **swlist** `l= MQ_INSTALLATION_PATH MQSERIES` to list all of the IBM MQ components that are installed.
7. Decide whether to install the updates interactively, and if you want to control which components are updated.

   You can update in the following ways:
   - Silently update all the installed IBM MQ components by installing the whole maintenance package.

     ```
     swinstall -s $PWD/service_update_package
     MQSERIES,l= MQ_INSTALLATION_PATH
     ```

     The **swinstall** command attempts to find an installed component for every component in the update package, and updates it. **swinstall** writes out error messages for components that it cannot find.
   - Silently update some IBM MQ components by installing only the required updates from the maintenance package.

     If you specify *update_components* correctly, the update procedure can be error-free. **swinstall** only updates components that you have listed and components that are dependent on components you have listed.

     a. Using the list of installed IBM MQ components, create a space separated list of the components you want to update (*update_components*). This list requires the installation path of each component to be specified, in the form: *component* `,l= MQ_INSTALLATION_PATH`

     b. ```
        swinstall -s $PWD/service_update_package
        update_components
        ```
   - Interactively update some IBM MQ components from the maintenance package, selecting only the update components that are required.

     ```
     swinstall -s $PWD/service_update_package
     ```

     a. Open **MQSERIES** and mark the update components you want to apply. Correctly marked, there are no errors when the updates are applied. The installer resolves dependencies automatically.

     b. Select **Actions > Change Product Location** to change the IBM MQ installation you intend to update.

     c. Select **Actions > Install**. The log file tells you if there are any problems that need fixing.
     ```

Related information:
dspmq
Stopping a queue manager
DISPLAY LSSTATUS
endmqm (end queue manager)
endmqlsr (end listener)
Applying maintenance level updates to multi-instance queue managers on UNIX and Linux

Reverting to the previous maintenance level on HP-UX

HP-UX

You revert to a previous maintenance level of IBM MQ by using **swremove**.

Before you begin

1. If you are running on a server with multiple IBM MQ installations, you must identify the installation. Make sure that the commands you enter run against the correct installation; see setmqenv.

You can apply and remove maintenance from a IBM MQ MQI client that is not installed on the same server as a queue manager. You do not have to stop any queue managers or logon as administrator. Because you do not have to stop any queue managers, do not do steps 1 to 3 in the following maintenance procedure.

Procedure

1. Log in as a user in `group mqm`.
2. Stop all applications using the IBM MQ installation.

   If you use the Managed File Transfer (MFT) component, ensure that any MFT agents have finished all of the file transfers that they were engaged in. There should be no incomplete transfers associated with the agents, and their SYSTEM.FTE.STATE queues should contain no messages.
3. End all the activity of queue managers associated with the IBM MQ installation.
   a. Run the **dspmq** command to list the state of all the queue managers on the system.

      Run either of the following commands from the installation that you are updating:
      ```
      dspmq -o installation -o status
      dspmq -a
      ```
      dspmq -o installation -o status displays the installation name and status of queue managers associated with all installations of IBM MQ.

      dspmq -a displays the status of active queue managers associated with the installation from which the command is run.
   b. Use the MQSC command **DISPLAY LSSTATUS** to list the status of listeners associated with a queue manager, as shown in the following example:
      ```
      echo "DISPLAY LSSTATUS(*) STATUS" | runmqsc QmgrName
      ```
   c. Run the **endmqm** command to stop each running queue manager associated with this installation.

```
              ┌─-c─┐
►►──endmqm────┼─-w─┼──QmgrName──────────────────────────────────────────────────►◄
              ├─-i─┤
              └─-p─┘
```

The **endmqm** command informs an application that the queue manager it is connected to is stopping; see Stopping a queue manager.

For the maintenance to proceed, applications must respond to an **endmqm** command by disconnecting from the queue manager and releasing any IBM MQ libraries they have loaded. If they do not, you must find another way to force applications to release IBM MQ resources, such as by stopping the applications.

You must also stop applications that are using the client libraries that are part of the installation. Client applications might be connected to a different queue manager, running a different installation of IBM MQ. The application is not informed about queue managers in the current installation being shut down.

Any applications that continue to have IBM MQ shared libraries from the installation loaded prevent you applying IBM MQ maintenance. An application might disconnect from a queue manager, or be forcibly disconnected, but keep an IBM MQ shared library loaded.

Note: "Applying maintenance level updates to multi-instance queue managers on UNIX and Linux" on page 611 describes how to apply maintenance to a multi-instance queue manager. A multi-instance queue manager can continue to run on one server, while maintenance is applied to another server.

d. Stop any listeners associated with the queue managers, using the command:

   ```
   endmqlsr -m QMgrName
   ```

4. Log in as root, or switch to the superuser using the **su** command.
5. Run the `swremove` command to remove the maintenance package from the system.

   For example, to remove the 7.R.0.1 maintenance level, use the command:

   ```
   swremove MQSERIES,r=7.R.0.1,l= MQ_INSTALLATION_PATH
   ```

   where:

   - `R` is the number of the Release
   - *MQ_INSTALLATION_PATH* is the installation path for IBM MQ

   Details of the **swremove** command can be found in the *HP-UX Administration Guide* or by using the **man swremove** command.

Related information:
dspmq
Stopping a queue manager
DISPLAY LSSTATUS
endmqm (end queue manager)
endmqlsr (end listener)
Applying maintenance level updates to multi-instance queue managers on UNIX and Linux

Applying maintenance level updates on Linux

Linux

You can apply maintenance level updates to IBM MQ on Linux by using RPM. The same procedure applies to all Linux platforms, including Ubuntu.

Before you begin

If you are running on a server with multiple IBM MQ installations, you must identify the installation. Make sure that the commands you enter run against the correct installation; see setmqenv.

About this task

Maintenance level updates are delivered in the form of Red Hat Package Manager (RPM) update images, which are applied using the RPM installation tool.

You can apply and remove maintenance from a IBM MQ MQI client that is not installed on the same server as a queue manager. You do not have to stop any queue managers or logon as administrator. Because you do not have to stop any queue managers, do not do steps 1 to 3 in the following maintenance procedure.

Important: **`pax`** and **`rpmbuild`** are not supplied as part of the product. You must obtain these from your Linux distribution supplier.

Additional disk space is required for the update images to allow maintenance level updates to be removed and the previous level restored. The updated files are kept in *`MQ_INSTALLATION_PATH`*`/maintenance` directory. Do not delete or move this directory or the files it contains.

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

Updates are cumulative. You can apply your chosen update directly, without applying any previous updates first. The maintenance level updates might contain updates for one or more packages. You must apply those parts of an update that correspond to the packages that are applied in your installation.

Important: Although it is possible to install a fix pack at the same level as an installation performed from a manufacturing refresh image at that level, you should not attempt this process. Installing a fix pack at the same level as the one already on your system, can leave the package management database of your system in an inconsistent state with respect to the installation of IBM MQ.

Procedure

1. Log in as a user in `group mqm`.
2. Stop all applications using the IBM MQ installation.

   If you use the Managed File Transfer (MFT) component, ensure that any MFT agents have finished all of the file transfers that they were engaged in. There should be no incomplete transfers associated with the agents, and their SYSTEM.FTE.STATE queues should contain no messages.

3. End all the activity of queue managers associated with the IBM MQ installation.
   a. Run the **dspmq** command to list the state of all the queue managers on the system.

      Run either of the following commands from the installation that you are updating:
      ```
      dspmq -o installation -o status
      dspmq -a
      ```
      dspmq -o installation -o status displays the installation name and status of queue managers associated with all installations of IBM MQ.

      dspmq -a displays the status of active queue managers associated with the installation from which the command is run.
   b. Use the MQSC command **DISPLAY LSSTATUS** to list the status of listeners associated with a queue manager, as shown in the following example:
      ```
      echo "DISPLAY LSSTATUS(*) STATUS" | runmqsc QmgrName
      ```
   c. Run the **endmqm** command to stop each running queue manager associated with this installation.
      ```
                 ┌-c─┐
      ►►─endmqm──┼-w─┼─QmgrName──────────────────────────────────────────►◄
                 ├-i─┤
                 └-p─┘
      ```
      The **endmqm** command informs an application that the queue manager it is connected to is stopping; see Stopping a queue manager.

      For the maintenance to proceed, applications must respond to an **endmqm** command by disconnecting from the queue manager and releasing any IBM MQ libraries they have loaded. If they do not, you must find another way to force applications to release IBM MQ resources, such as by stopping the applications.

      You must also stop applications that are using the client libraries that are part of the installation. Client applications might be connected to a different queue manager, running a different installation of IBM MQ. The application is not informed about queue managers in the current installation being shut down.

      Any applications that continue to have IBM MQ shared libraries from the installation loaded prevent you applying IBM MQ maintenance. An application might disconnect from a queue manager, or be forcibly disconnected, but keep an IBM MQ shared library loaded.

      Note: "Applying maintenance level updates to multi-instance queue managers on UNIX and Linux" on page 611 describes how to apply maintenance to a multi-instance queue manager. A multi-instance queue manager can continue to run on one server, while maintenance is applied to another server.
   d. Stop any listeners associated with the queue managers, using the command:
      ```
      endmqlsr -m QMgrName
      ```
4. Log in as root, or switch to the superuser using the **su** command.
5. Change into the directory containing the maintenance packages.
6. Run the `ls` command to list the available updates.

   For example, if there are level 1 maintenance updates for the Runtime, SDK and Server packages, you see the following:
   ```
   MQSeriesRuntime-Uxxxx-V.R.0-1.i386.rpm
   MQSeriesSDK-Uxxxx-V.R.0-1.i386.rpm
   MQSeriesServer-Uxxxx-V.R.0-1.i386.rpm
   ```
   where `V` is the version number and `R` is the number of the Release.
7. Run the **rpm** command to find out which packages are installed on your server.

   Enter the following command:
   ```
   rpm -qa | grep MQSeries
   ```

Note: If you are using Ubuntu, add the **--force-debian** attribute.

```
rpm --force-debian -qa | grep MQSeries
```

For example, if you have a minimum IBM MQ installation and SDK component, at level 0, the **rpm** command returns:

```
MQSeriesRuntime-V.R.0-0
MQSeriesSDK-V.R.0-0
MQSeriesServer-V.R.0-0
```

where `V` is the version number and `R` is the number of the Release.

8. If this fix pack is to be upgraded on an installation, other than the first installation on the system, run the **crtmqfp** command to create and use a unique set of packages to install on the system. Note, that if this is the first, or only, IBM MQ installation on the system, you can ignore this step. You must install the **pax** command in order for the **crtmqfp** command to run on Linux.
   a. Run the command `./crtmqfp` *suffixname* where *suffixname* is the same as the suffix used during renaming of the base level IBM MQ installation.
   b. Set your current directory to the location specified when the **crtmqfp** command completes. This directory is a subdirectory of `/var/tmp/mq_rpms`, in which the unique set of packages is created. The packages have the suffix value contained within the filename.

      For example, if you used suffix 1 during repackaging of the base level IBM MQ installation, enter the command: `./crtmqfp 1`.

      There is now a subdirectory named `/var/tmp/mq_rpms/1/xxxx`, and the packages will be renamed, for example, from `MQSeriesRuntime-V.R.0-1.xxxx.rpm` to `MQSeriesRuntime_1-V.R.0-1.xxxx.rpm`. Where `V` is the version number and `R` is the number of the Release.
9. Run the **rpm** command to apply all available updates for the packages you have on your system:
   - To update an installation in the default location, `/opt/mqm`:

     ```
     rpm -ivh MQSeriesRuntime-Uxxxx-V.R.0-1.i386.rpm
       MQSeriesSDK-Uxxxx-V.R.0-1.i386.rpm
       MQSeriesServer-Uxxxx-V.R.0-1.i386.rpm
     ```

     where `V` is the version number and `R` is the number of the Release.
   - To update an installation in a custom location, specify the **rpm** `prefix` option:

     ```
     rpm --prefix /opt/customLocation -ivh MQSeriesRuntime-Uxxxx-V.R.0-1.i386.rpm
       MQSeriesSDK-Uxxxx-V.R.0-1.i386.rpm
       MQSeriesServer-Uxxxx-V.R.0-1.i386.rpm
     ```

     where `V` is the version number and `R` is the number of the Release.

   You must apply all packages in a maintenance update that correspond to those packages that are currently installed on your system.
10. Repeat step 7 on page 597 to list the packages that are now available.

    The Runtime, SDK, and Server packages are now at level 1:

    ```
    MQSeriesRuntime-V.R.0-0
    MQSeriesSDK-V.R.0-0
    MQSeriesServer-V.R.0-0
    MQSeriesRuntime-Uxxxx-V.R.0-1
    MQSeriesSDK-Uxxxx-V.R.0-1
    MQSeriesServer-Uxxxx-V.R.0-1
    ```

    where `V` is the version number and `R` is the number of the Release.

    Note:

After the installation of IBM MQ fix packs, if you run the `rpm-verify` or `rpm -V` command, it does not return the correct results. It produces spurious results relating to missing files in *MQ_INSTALLATION_PATH*/maintenance.

This error message can be ignored because it is a known limitation in the IBM MQ fix pack installation code. For further information about this error, see IBM MQ Fix Pack installation errors - Linux reports errors

What to do next

For further information about using RPM to install software packages, see your Linux documentation.

Related information:

dspmq

Stopping a queue manager

DISPLAY LSSTATUS

endmqm (end queue manager)

endmqlsr (end listener)

Applying maintenance level updates to multi-instance queue managers on UNIX and Linux

Reverting to the previous maintenance level on Linux

Linux

You can remove updates and revert to the previous maintenance level of IBM MQ by using **RPM**. The same procedure applies to all Linux platforms, including Ubuntu.

Before you begin

If you are running on a server with multiple IBM MQ installations, you must identify the installation. Make sure that the commands you enter run against the correct installation; see setmqenv.

About this task

When maintenance is applied, the original versions of replaced files are saved to allow the updates to be removed if necessary. To restore the previous maintenance level, run an Red Hat Package Manager, RPM, uninstall command for all the packages that were updated by the maintenance package as follows:

Procedure

1. Log in as a user in `group mqm`.
2. Stop all applications using the IBM MQ installation.

   If you use the Managed File Transfer (MFT) component, ensure that any MFT agents have finished all of the file transfers that they were engaged in. There should be no incomplete transfers associated with the agents, and their SYSTEM.FTE.STATE queues should contain no messages.
3. End all the activity of queue managers associated with the IBM MQ installation.
   a. Run the **dspmq** command to list the state of all the queue managers on the system.

      Run either of the following commands from the installation that you are updating:

      ```
      dspmq -o installation -o status
      dspmq -a
      ```

      dspmq -o installation -o status displays the installation name and status of queue managers associated with all installations of IBM MQ.

      dspmq -a displays the status of active queue managers associated with the installation from which the command is run.

b. Use the MQSC command **DISPLAY LSSTATUS** to list the status of listeners associated with a queue manager, as shown in the following example:

```
echo "DISPLAY LSSTATUS(*) STATUS" | runmqsc QmgrName
```

c. Run the **endmqm** command to stop each running queue manager associated with this installation.

```
               ┌─-c─┐
►►──endmqm─────┼─-w─┼──QmgrName──────────────────────────────────────────────────►◄
               ├─-i─┤
               └─-p─┘
```

The **endmqm** command informs an application that the queue manager it is connected to is stopping; see Stopping a queue manager.

For the maintenance to proceed, applications must respond to an **endmqm** command by disconnecting from the queue manager and releasing any IBM MQ libraries they have loaded. If they do not, you must find another way to force applications to release IBM MQ resources, such as by stopping the applications.

You must also stop applications that are using the client libraries that are part of the installation. Client applications might be connected to a different queue manager, running a different installation of IBM MQ. The application is not informed about queue managers in the current installation being shut down.

Any applications that continue to have IBM MQ shared libraries from the installation loaded prevent you applying IBM MQ maintenance. An application might disconnect from a queue manager, or be forcibly disconnected, but keep an IBM MQ shared library loaded.

Note: "Applying maintenance level updates to multi-instance queue managers on UNIX and Linux" on page 611 describes how to apply maintenance to a multi-instance queue manager. A multi-instance queue manager can continue to run on one server, while maintenance is applied to another server.

d. Stop any listeners associated with the queue managers, using the command:

```
endmqlsr -m QMgrName
```

4. Log in as root, or switch to the superuser using the **su** command.
5. Run the **rpm** command to find out which packages are installed on your server.

   Enter the following command:

```
rpm -qa | grep MQSeries
```

Note: If you are using Ubuntu, add the **--force-debian** attribute.

```
rpm --force-debian -qa | grep MQSeries
```

Using the example given in "Applying maintenance level updates on Linux" on page 596, returns:

```
MQSeriesRuntime-V.R.0-0
MQSeriesSDK-V.R.0-0
MQSeriesServer-V.R.0-0
MQSeriesRuntime-Uxxxx-V.R.0-1
MQSeriesSDK-Uxxxx-V.R.0-1
MQSeriesServer-Uxxxx-V.R.0-1
```

where V is the version number and R is the number of the Release.

6. Run the **rpm** command to remove all the updates applied at level 1.

   Enter the following commands:

```
rpm -ev MQSeriesRuntime-Uxxxx-V.R.0-1 MQSeriesSDK-Uxxxx-V.R.0-1
MQSeriesServer-Uxxxx-V.R.0-1
```

where V is the version number and R is the number of the Release.

7. Repeat step 5 to check that the ptf packages have been removed, leaving only the original installation packages:

   ```
   MQSeriesRuntime-V.R.0-0
   MQSeriesSDK-V.R.0-0
   MQSeriesServer-V.R.0-0
   ```

   where `V` is the version number and `R` is the number of the Release.

What to do next

For further information on using RPM to install software packages, see your Linux documentation.

Related information:
dspmq
Stopping a queue manager
DISPLAY LSSTATUS
endmqm (end queue manager)
endmqlsr (end listener)
Applying maintenance level updates to multi-instance queue managers on UNIX and Linux

Applying maintenance level updates on IBM MQ on Solaris

Solaris

You can apply maintenance level updates to IBM MQ for Solaris using **`pkgadd`**.

Before you begin

1. Ensure that you have enough disk space to apply maintenance level updates. A maintenance level update requires hard disk space for installation. In addition, the installation process might require a similar amount of disk space to save the previous level. For example, a 16 MB update might require 32 MB of space. The additional space allows a maintenance level update to be removed, and the previous level to be restored automatically.
2. If you are running on a server with multiple IBM MQ installations, you must identify the installation. Make sure that the commands you enter run against the correct installation; see setmqenv.

You can apply and remove maintenance from a IBM MQ MQI client that is not installed on the same server as a queue manager. You do not have to stop any queue managers or logon as administrator. Because you do not have to stop any queue managers, do not do steps 1 to 3 in the following maintenance procedure.

About this task

Stop applications using the installation and use **`pkgadd`** to install maintenance.

Important: Although it is possible to install a fix pack at the same level as an installation performed from a manufacturing refresh image at that level, you should not attempt this process. Installing a fix pack at the same level as the one already on your system, can leave the package management database of your system in an inconsistent state with respect to the installation of IBM MQ.

Procedure

1. Log in as a user in `group mqm`.
2. Stop all applications using the IBM MQ installation.

   If you use the Managed File Transfer (MFT) component, ensure that any MFT agents have finished all of the file transfers that they were engaged in. There should be no incomplete transfers associated with the agents, and their SYSTEM.FTE.STATE queues should contain no messages.

3. End all the activity of queue managers associated with the IBM MQ installation.
   a. Run the **dspmq** command to list the state of all the queue managers on the system.

      Run either of the following commands from the installation that you are updating:

      ```
      dspmq -o installation -o status
      dspmq -a
      ```

      dspmq -o installation -o status displays the installation name and status of queue managers associated with all installations of IBM MQ.

      dspmq -a displays the status of active queue managers associated with the installation from which the command is run.
   b. Use the MQSC command **DISPLAY LSSTATUS** to list the status of listeners associated with a queue manager, as shown in the following example:

      ```
      echo "DISPLAY LSSTATUS(*) STATUS" | runmqsc QmgrName
      ```
   c. Run the **endmqm** command to stop each running queue manager associated with this installation.

      ```
                   ┌─-c─┐
      ►►──endmqm───┼─-w─┼──QmgrName──────────────────────────────────────►◄
                   ├─-i─┤
                   └─-p─┘
      ```

      The **endmqm** command informs an application that the queue manager it is connected to is stopping; see Stopping a queue manager.

      For the maintenance to proceed, applications must respond to an **endmqm** command by disconnecting from the queue manager and releasing any IBM MQ libraries they have loaded. If they do not, you must find another way to force applications to release IBM MQ resources, such as by stopping the applications.

      You must also stop applications that are using the client libraries that are part of the installation. Client applications might be connected to a different queue manager, running a different installation of IBM MQ. The application is not informed about queue managers in the current installation being shut down.

      Any applications that continue to have IBM MQ shared libraries from the installation loaded prevent you applying IBM MQ maintenance. An application might disconnect from a queue manager, or be forcibly disconnected, but keep an IBM MQ shared library loaded.

      Note: "Applying maintenance level updates to multi-instance queue managers on UNIX and Linux" on page 611 describes how to apply maintenance to a multi-instance queue manager. A multi-instance queue manager can continue to run on one server, while maintenance is applied to another server.
   d. Stop any listeners associated with the queue managers, using the command:

      ```
      endmqlsr -m QMgrName
      ```
4. Log in as root, or switch to the superuser using the **su** command.
5. Change into the directory containing the maintenance packages.
6. Run the **crtmqfp** command to create and use a unique set of packages to install on the system, if this fix pack is to be upgraded on a installation that is not the first installation on the system. This command creates and uses a unique set of packages to install on the system.
   a. Run the command **crtmqfp** mqm- *suffixname* where *suffixname* is the same as the suffix used during renaming of the base level IBM MQ installation. Note that this command creates a full copy of the installation packages in a subdirectory of `/var/tmp`.
   b. Set your current directory to the location specified when the **crtmqfp** command completes. This directory is a subdirectory of `/var/spool`, in which the unique set of packages is created. The packages have the suffix value contained within the filename.
7. Proceed with installation using the following command: Enter the following command to start the installation process if this fix pack is to be upgraded on an installation that is

a. The first installation on the system:

   ```
   pkgadd -d packagename
   ```

   where `packagename` corresponds to the image file name. For example:

   ```
   mqm-U1234.img
   ```
b. Not the first installation on the system:

   ```
   pkgadd mqm-suffixname
   ```

   where `suffixname` is the name of the directory created in `/var/spool/pkg`.

   For example, if you install IBM WebSphere MQ Version 7.0 as a package called `mqm-main7` and create a package to upgrade to IBM WebSphere MQ Version 7.0.0.1, using the command **crtmqfp** `mqm-main7`, package *mqm-main7-07-00-00-01* is created in `/var/spool/pkg`.

   To install package *mqm-main7-07-00-00-01*, issue the command **pkgadd** `mqm-main7-07-00-00-01`.

For further information about using **pkgadd** to install software packages, see the Solaris documentation.

8. Follow the on-screen instructions.

Related information:

dspmq

Stopping a queue manager

DISPLAY LSSTATUS

endmqm (end queue manager)

endmqlsr (end listener)

Applying maintenance level updates to multi-instance queue managers on UNIX and Linux

Applying maintenance level updates in non-interactive mode on Solaris

You can install IBM MQ for Solaris non-interactively by creating a response file and an admin file.

Before you begin

1. Ensure that you have enough disk space to apply maintenance level updates. A maintenance level update requires hard disk space for installation. In addition, the installation process might require a similar amount of disk space to save the previous level. For example, a 16 MB update might require 32 MB of space. The additional space allows a maintenance level update to be removed, and the previous level to be restored automatically.
2. If you are running on a server with multiple IBM MQ installations, you must identify the installation. Make sure that the commands you enter run against the correct installation; see setmqenv.

You can apply and remove maintenance from a IBM MQ MQI client that is not installed on the same server as a queue manager. You do not have to stop any queue managers or logon as administrator. Because you do not have to stop any queue managers, do not do steps 1 to 3 in the following maintenance procedure.

About this task

Stop applications using the installation and use **pkgadd** to install maintenance.

Important: Although it is possible to install a fix pack at the same level as an installation performed from a manufacturing refresh image at that level, you should not attempt this process. Installing a fix pack at the same level as the one already on your system, can leave the package management database of your system in an inconsistent state with respect to the installation of IBM MQ.

Procedure

1. Log in as a user in `group mqm`.
2. Stop all applications using the IBM MQ installation.

   If you use the Managed File Transfer (MFT) component, ensure that any MFT agents have finished all of the file transfers that they were engaged in. There should be no incomplete transfers associated with the agents, and their SYSTEM.FTE.STATE queues should contain no messages.
3. End all the activity of queue managers associated with the IBM MQ installation.
   a. Run the **dspmq** command to list the state of all the queue managers on the system.

      Run either of the following commands from the installation that you are updating:

      ```
      dspmq -o installation -o status
      dspmq -a
      ```

      dspmq -o installation -o status displays the installation name and status of queue managers associated with all installations of IBM MQ.

      dspmq -a displays the status of active queue managers associated with the installation from which the command is run.
   b. Use the MQSC command **DISPLAY LSSTATUS** to list the status of listeners associated with a queue manager, as shown in the following example:

      ```
      echo "DISPLAY LSSTATUS(*) STATUS" | runmqsc QmgrName
      ```
   c. Run the **endmqm** command to stop each running queue manager associated with this installation.

      ```
                 ┌-c┐
      ►►─endmqm──┼-w┼──QmgrName─────────────────────────────────────────►◄
                 ├-i┤
                 └-p┘
      ```

      The **endmqm** command informs an application that the queue manager it is connected to is stopping; see Stopping a queue manager.

      For the maintenance to proceed, applications must respond to an **endmqm** command by disconnecting from the queue manager and releasing any IBM MQ libraries they have loaded. If they do not, you must find another way to force applications to release IBM MQ resources, such as by stopping the applications.

      You must also stop applications that are using the client libraries that are part of the installation. Client applications might be connected to a different queue manager, running a different installation of IBM MQ. The application is not informed about queue managers in the current installation being shut down.

      Any applications that continue to have IBM MQ shared libraries from the installation loaded prevent you applying IBM MQ maintenance. An application might disconnect from a queue manager, or be forcibly disconnected, but keep an IBM MQ shared library loaded.

      Note: "Applying maintenance level updates to multi-instance queue managers on UNIX and Linux" on page 611 describes how to apply maintenance to a multi-instance queue manager. A multi-instance queue manager can continue to run on one server, while maintenance is applied to another server.
   d. Stop any listeners associated with the queue managers, using the command:

      ```
      endmqlsr -m QMgrName
      ```
4. Log in as root, or switch to the superuser using the **su** command.
5. Change into the directory containing the maintenance packages.
6. Run the **crtmqfp** command to create and use a unique set of packages to install on the system, if this fix pack is to be upgraded on a installation that is not the first installation on the system. This command creates and uses a unique set of packages to install on the system.

   a. Run the command **crtmqfp** mqm- *suffixname* where *suffixname* is the same as the suffix used during renaming of the base level IBM MQ installation. Note that this command creates a full copy of the installation packages in a subdirectory of `/var/tmp`.
   b. Set your current directory to the location specified when the **crtmqfp** command completes. This directory is a subdirectory of `/var/spool`, in which the unique set of packages is created. The packages have the suffix value contained within the filename.
7. Create the non-interactive installation response file using the **pkgask** command. Enter the following command to create the response file if this fix pack is to be upgraded on an installation that is:
   a. The first installation on the system:
      ```
      pkgask -d location_to_image/imagefile -r response.txt packagename
      ```
      where *imagefile* corresponds to the image file name, for example `mqm-U200403.img`, `response.txt` is the name of the response file to create, and *packagename* is the fix pack package name, for example `mqm-07-05-00-02`.
   b. Not the first installation on the system:
      ```
      pkgask -d /var/spool/pkg -r response.txt mqm-suffixname
      ```
      where `/var/spool/pkg` is the location of the new package, `response.txt` is the name of the response file to create, and *suffixname* is the name of the directory created in `/var/spool/pkg`.
8. Find the `admin_file` from the server installation media located at *install_media*/silent/admin or create an `admin_file` in the following format:
   ```
   mail=
   instance=unique
   partial=ask
   runlevel=ask
   idepend=ask
   rdepend=ask
   space=ask
   setuid=nocheck
   conflict=nocheck
   action=nocheck
   basedir=default
   ```
9. Run the **pkgadd** command to apply the maintenance level update IBM MQ for Solaris in non-interactive mode. Enter the following command to start the installation process if this fix pack is to be upgraded on an installation that is:
   a. The first installation on the system:
      ```
      pkgadd -v -n -r response.txt -a admin_file -d location_to_image/imagefile packagename
      ```
      where *admin_file* is a path qualified name of the admin file you created, and *packagename* corresponds to the fix pack package being installed.
   b. Not the first installation on the system:
      ```
      pkgadd -v -n -r response.txt -a admin_file -d /var/spool/pkg mqm-suffixname
      ```
10. Follow the on-screen instructions.

Related information:
dspmq
Stopping a queue manager
DISPLAY LSSTATUS
endmqm (end queue manager)
endmqlsr (end listener)
Applying maintenance level updates to multi-instance queue managers on UNIX and Linux

Reverting to the previous maintenance level on Solaris

Solaris

You can revert to a previous maintenance level of IBM MQ by stopping IBM MQ and using **pkgrm**.

Before you begin

If you are running on a server with multiple IBM MQ installations, you must identify the installation. Make sure that the commands you enter run against the correct installation; see setmqenv.

About this task

When maintenance is applied, the original versions of replaced files are saved to allow the updates to be removed if necessary. To restore the previous maintenance level, run **pkgrm** command for all the packages that were updated by the maintenance package as follows:

Procedure

1. Log in as a user in `group mqm`.
2. Stop all applications using the IBM MQ installation.

   If you use the Managed File Transfer (MFT) component, ensure that any MFT agents have finished all of the file transfers that they were engaged in. There should be no incomplete transfers associated with the agents, and their SYSTEM.FTE.STATE queues should contain no messages.
3. End all the activity of queue managers associated with the IBM MQ installation.
   a. Run the **dspmq** command to list the state of all the queue managers on the system.

      Run either of the following commands from the installation that you are updating:
      ```
      dspmq -o installation -o status
      dspmq -a
      ```
      dspmq -o installation -o status displays the installation name and status of queue managers associated with all installations of IBM MQ.

      dspmq -a displays the status of active queue managers associated with the installation from which the command is run.
   b. Use the MQSC command **DISPLAY LSSTATUS** to list the status of listeners associated with a queue manager, as shown in the following example:
      ```
      echo "DISPLAY LSSTATUS(*) STATUS" | runmqsc QmgrName
      ```
   c. Run the **endmqm** command to stop each running queue manager associated with this installation.

```
            ┌─-c─┐
►►──endmqm──┼─-w─┼──QmgrName──────────────────────────────────────────►◄
            ├─-i─┤
            └─-p─┘
```

The **endmqm** command informs an application that the queue manager it is connected to is stopping; see Stopping a queue manager.

For the maintenance to proceed, applications must respond to an **endmqm** command by disconnecting from the queue manager and releasing any IBM MQ libraries they have loaded. If they do not, you must find another way to force applications to release IBM MQ resources, such as by stopping the applications.

You must also stop applications that are using the client libraries that are part of the installation. Client applications might be connected to a different queue manager, running a different installation of IBM MQ. The application is not informed about queue managers in the current installation being shut down.

Any applications that continue to have IBM MQ shared libraries from the installation loaded prevent you applying IBM MQ maintenance. An application might disconnect from a queue manager, or be forcibly disconnected, but keep an IBM MQ shared library loaded.

Note: "Applying maintenance level updates to multi-instance queue managers on UNIX and Linux" on page 611 describes how to apply maintenance to a multi-instance queue manager. A multi-instance queue manager can continue to run on one server, while maintenance is applied to another server.

d. Stop any listeners associated with the queue managers, using the command:

```
endmqlsr -m QMgrName
```

4. Log in as root, or switch to the superuser using the **su** command.
5. Run the **pkgrm** command to remove the latest maintenance update from the system:

```
pkgrm packagename
```

packagename is the name of the package that you want to remove; for example, `mqm-07-R-00-01`, where `R` is the number of the Release.

Details of the **pkgrm** command can be found in the Solaris documentation, or by using the **man pkgrm** command.

If you do not know the name of the package to remove, try listing the packages that are installed using the following command: `pkginfo | grep mqm`

Note: Ignore any error messages of the form `shared pathname not removed`.

What to do next

If you have installed an IBM MQ MQI client, and the client was updated after installing the maintenance level that is being removed, you must specifically update your IBM MQ MQI client installation again, after the maintenance level has been removed

Related information:
dspmq
Stopping a queue manager
DISPLAY LSSTATUS
endmqm (end queue manager)
endmqlsr (end listener)
Applying maintenance level updates to multi-instance queue managers on UNIX and Linux

Staging maintenance fixes on UNIX and Linux

UNIX Linux

On UNIX and Linux, you can use multiple installations of IBM MQ on the same server to control the release of maintenance fixes.

Before you begin

Set up your configuration modeled on the first row of Figure 60 on page 609. You can apply this scenario to any version of IBM MQ from IBM WebSphere MQ Version 7.1 onwards. In this scenario it is assumed you have a number of applications and two queue managers, `QM1` and `QM2`, running on a server. IBM WebSphere MQ Version 7.0.1 is not installed on the server.

1. Install two copies of IBM MQ. In the example, they are named `Inst_1` and `Inst_2` and IBM WebSphere MQ Version 7.1 is being used.
2. Make `Inst_1` primary by running **`setmqinst`**.
3. Associate all the queue managers on the server with `Inst_1` by running **`setmqm`**.
4. Start all the queue managers on the server.
5. Show and connect all direct connections with the queue managers associated with `Inst_1` in IBM MQ Explorer.
6. Set up remote connections to all the queue managers in each instance of IBM MQ Explorer.

About this task

You can install multiple copies of IBM MQ on a server to stage the release of IBM MQ fixes. Figure 60 on page 609 illustrates a way of using two installations to roll out fixes. In this approach, you maintain two fix levels on a server, with the aim of getting all queue managers and applications to the production fix level before replacing the previous level on fix pack with the next level.

Which installation an application uses is driven by the queue manager an application connects to. The **`setmqm`** command associates a queue manager with an installation. You can associate a queue manager with a different installation as long as the installation is at the same or higher command level. In this example, all the installations are at the same command level. You can associate or reassociate a queue manager with either of the installations running any of the fix packs.

In the example, an application links to the primary installation. When it connects to a queue manager, IBM MQ switches the linkage to the installation associated with the queue manager; see "Multi-installation queue manager coexistence on UNIX, Linux, and Windows" on page 668.

For applications built with the link options described in the product documentation, the simplest way to configure the link library search path for IBM MQ applications is to make an installation primary. Only if it is important to pick up a fix in the IBM MQ link library itself, must you review the search path. Either you must make the installation with the IBM MQ link library fix primary, or make a local adjustment for the application, perhaps by running the **`setmqenv`** command.

Running commands is a different matter. Commands are always run from the primary installation, or the installation you have selected by running the **setmqenv** command. If you run a command from the wrong installation, the command fails. For example, if QM1 is associated with Inst_1, running the Windows command, Inst_2_Installation_path/bin/strmqm QM1 fails.

If you are using IBM MQ Explorer and you have two installations, you also have two IBM MQ Explorer instances. One linked to one installation, and one to the other. Each IBM MQ Explorer shows locally connected queue managers that are associated with the same installation as the instance of IBM MQ Explorer. To monitor all the queue managers on a server, set up remote connections to the queue managers associated with the other installations.

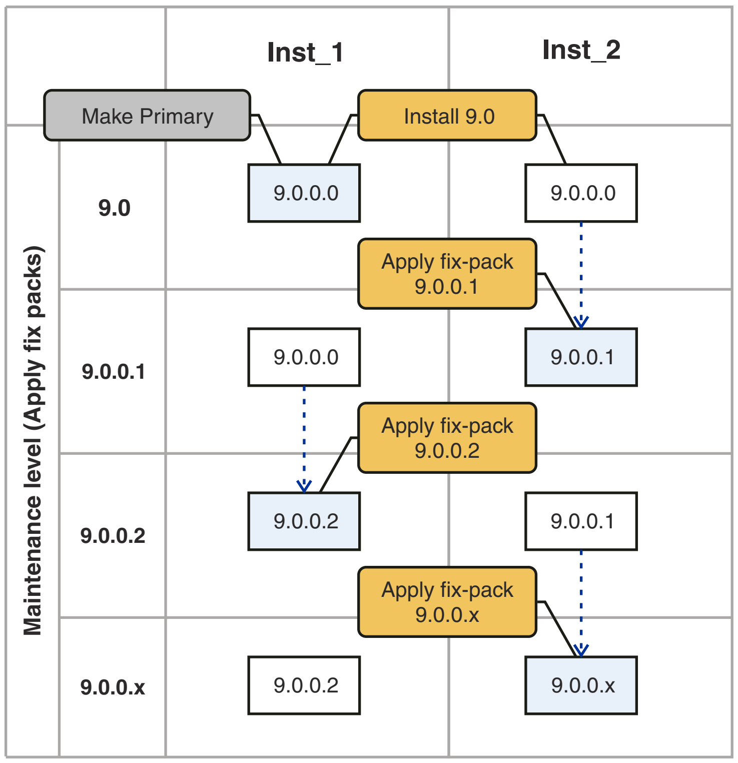


Figure 60. Rolling fix releases

Procedure

1. Download the first fix pack, for example, 7.1.0.1, when it is released.

   See Fix Central.
2. Apply the fix pack you downloaded to Inst_2. See "Applying and removing maintenance on UNIX and Linux" on page 586.
3. Verify Inst_2.
4. Transfer the queue managers to Inst_2 one at a time.
   a. Stop QM1 and the applications connected to it.

      The **endmqm** command informs an application that the queue manager it is connected to is stopping; see Stopping a queue manager.

      Note: The topic, "Applying maintenance level updates to multi-instance queue managers on Windows" on page 577, describes how to apply maintenance to a multi-instance queue manager. A multi-instance queue manager can continue to run on one server, while maintenance is applied to another server.

b. Set up the local environment to the installation `Inst_2`.

```
. Inst_2_INSTALLATION_PATH/bin/setmqenv -s
```

c. Associate the queue manager with `Inst_2`.

```
setmqm -m QM1 -n Inst_2
```

d. Start `QM1`

```
strmqm QM1
```

e. Repeat substeps c and d for `QM2`.

f. Set up IBM MQ Explorer for `Inst_2`.

1) Start the `Inst_2` instance of IBM MQ Explorer

Tip: On Windows, hover over the IBM MQ icon in the system tray. The hover help shows the installation name associated with the IBM MQ Explorer instance.

2) Click **IBM MQ** > **Queue Managers** > **Show/Hide Queue Managers...** >

3) Click each directly connected queue manager listed in the Hidden Queue Managers list > **Show**.

4) Click **Close**.

5. Set `Inst_2` primary.

```
Inst_2_INSTALLATION_PATH/bin/setmqinst -i -n Inst_2
```

6. Download the next fix pack for the version of your product, for example, 7.1.0.2, when it is released.

   See Fix Central.
7. Apply the fix pack that you have just downloaded to `Inst_1`. See “Applying and removing maintenance on UNIX and Linux” on page 586.
8. Verify `Inst_1`.
9. Transfer queue managers to `Inst_1` one at a time.

   a. Follow the procedure in step 4 on page 609

      Replacing `Inst_2` by `Inst_1` in the instructions.
10. Set `Inst_1` primary.

```
Inst_1_INSTALLATION_PATH/bin/setmqinst -i -n Inst_1
```

11. Repeat steps 1 on page 609 to 5 for the odd-numbered fix packs of your product.
12. Repeat steps 6 to 10 for the even-numbered fix packs of your product.

Related concepts:

"Queue manager coexistence" on page 664
Queue managers, with different names, can coexist on any server as long as they use the same IBM MQ installation. On z/OS, UNIX, Linux, and Windows, different queue managers can coexist on the same server and be associated with different installations.

"Multi-installation queue manager coexistence on UNIX, Linux, and Windows" on page 668
You can install multiple copies of IBM MQ for UNIX, Linux, and Windows on the same server. The installations must be at Version 7.1 or later, with one exception. One Version 7.0.1 installation, at fix pack level 6, or later, can coexist with multiple Version 7.1, or later installations.

Related tasks:

"Migrating IBM MQ library loading from an earlier version of the product to the latest version" on page 684
No change in the way IBM MQ libraries are loaded is normally required if you upgrade from an earlier version of the product to the latest version. You must have followed the instructions on building IBM MQ applications in the earlier version, and you must replace the earlier version with the latest version of the product. If you choose to take advantage of multi-installation in the latest version of the product, based on the side-by-side or multi-stage migration scenarios, you must modify the environment for the operating system to resolve IBM MQ dependencies for an application. Typically, you can modify the runtime environment, rather than relink the application.

Related information:

Installing IBM MQ server on Windows

Associating a queue manager with an installation

Changing the primary installation

setmqenv

setmqinst

setmqm

Applying maintenance level updates to multi-instance queue managers on UNIX and Linux

Linux UNIX

On UNIX and Linux, you can use multi-instance queue managers to reduce the outage caused by applying maintenance updates.

Before you begin

Before starting this task, read through the prerequisites described in *Before you begin* in "Applying maintenance level server updates on Windows" on page 573

Before starting this task, see the Maintenance is applied to the IBM MQ installation on a server and not to individual queue managers. Before you apply maintenance, you must stop all the queue managers, and any IBM MQ service, on a server.

If you want a queue manager to keep running while maintenance is applied, you must configure it as a multi-instance queue manager, and have a standby instance running on another server. If the queue manager that you want to keep running is an existing single instance queue manager, you must convert it to a multi-instance queue manager. For prerequisites and guidance how to create a multi-instance queue manager, see Multi-instance queue managers.

You can create a multi-instance queue manager from Version 7.0.1 onwards. If you are running multi-instance queue managers, you then can apply a maintenance update to a running queue manager by switching the active instance to a different server.

Typically, active and standby installations are maintained at the same maintenance level. Consult the maintenance instructions for each update. Consult the instructions to see if it is possible to run the active and standby instances at different maintenance levels. Check whether fail over from higher to lower, or only lower to higher maintenance level is possible.

The instructions for applying a maintenance update might require you to stop a multi-instance queue manager completely.

If you have a primary server for running active queue manager instances, and a secondary server that runs standby instances, you have a choice of updating the primary or secondary server first. If you update the secondary server first, you must switch back to the primary server when both servers have been updated.

If you have active and standby instances on several servers, you must plan in what order you update the servers to minimize the disruption caused by ending the active instances on each server you update.

About this task

Combine the steps in this task with the maintenance update procedure for applying maintenance to an IBM MQ server installation.

Procedure

1. Where the maintenance update procedure instructs you to stop all running queue managers, or quiesce IBM MQ do the following instead: See: "Applying and removing maintenance on UNIX and Linux" on page 586
   a. If the queue manager is running as standby:
      - End the standby with the **endmqm** -x *QMgrName* command.
   b. If the queue manager is running as the active instance: End the instance and transfer control to the standby instance with the **endmqm** command. For example, **endmqm** *-shutdown_option* -s *QMgrName*, where *-shutdown_option* is an optional parameter specifying the type of shutdown. For more information, see endmqm.

      If there is no standby instance running, the command fails, and you must start a standby instance on a different server.
   c. If a queue manager is running as a single instance queue manager, you have no alternative but to stop the queue manager before applying the maintenance update.

   When you complete this step, no queue manager instances are left running on the server you intend to update.
2. Continue with the maintenance update procedure, following the step to issue the **endmqm** command, or quiesce IBM MQ and apply maintenance to the IBM MQ server.
3. When you have completed the maintenance update, restart all the queue managers on the IBM MQ server, permitting standby instances: Use the following command:
   ```
   strmqm -x QmgrName
   ```
4. Repeat the procedure on the standby server, to update its maintenance level.
5. If necessary, switch the active instances back to the primary servers: Use the **endmqm** *-shutdown_option* -s *QMgrName* command, and the restart the instances using the **strmqm** -x *QmgrName* command.

Applying and removing maintenance on IBM i

IBM i

Maintenance tasks associated with IBM i platforms are grouped in this section.

Procedure

- To apply maintenance level updates, see "Applying maintenance level updates on IBM i."
- To restore a queue manager to the previous version of the product from the latest version, see "Restoring a queue manager to a previous release on IBM i" on page 615.
- For information on how to use use multi-instance queue managers to reduce the outage caused by applying maintenance updates, see "Applying maintenance updates to multi-instance queue managers on IBM i" on page 616.

Related tasks:

"IBM MQ maintenance tasks on Multiplatforms - general information" on page 572
When you apply and remove maintenance level updates to IBM MQ, no migration is required. Maintenance level updates are applied either as a fix pack, or by manually applying an interim fix.

Applying maintenance level updates on IBM i

IBM i

You apply maintenance level updates on the latest release by stopping IBM MQ and using the IBM i standard maintenance procedure.

Before you begin

To find out what version you have currently installed, use the following commands:

Table 78. IBM MQ commands to display the installed versions

IBM MQ Product	Version command
IBM MQ Server	`DSPMQMVER`

IBM MQ Java	IBM MQ classes for Java:
	`java com.ibm.mq.MQJavaLevel`
	
	Note: For this command to work, you might need to set your environment classpath to include:
	• `/QIBM/ProdData/mqm/java/lib/com.ibm.mq.jar`
	
	IBM MQ classes for Java Message Service:
	`java com.ibm.mq.jms.MQJMSLevel`
	
	Note: For this command to work, you might need to set your environment classpath to include:
	• `/QIBM/ProdData/mqm/java/lib/com.ibm.mqjms.jar`
	
	See Environment variables relevant to IBM MQ classes for Java and Environment variables relevant to IBM MQ classes for JMS.

IBM MQ Client	`DSPMQMVER`

About this task

Maintenance updates for IBM i are supplied as PTFs (Program Temporary Fixes). They are available for download from the web as save files, which are normally stored in the QGPL library. IBM i PTF's can be found in "Fix Central" at the following location:

FixCentral.

Procedure

1. Read the cover letter carefully to see if you need to take any special actions.
2. Sign on to a new interactive IBM i session, ensuring that you are not accessing any IBM MQ objects.
3. Ensure that you have:
   a. `*ALLOBJ` authority, or object management authority for the `QMQM` library.
   b. Sufficient authority to use the ENDSBS command.
4. Warn all users that you are going to stop IBM MQ.
5. Use the ENDMQM command to quiesce all queue managers:
   ```
   ENDMQM MQMNAME(*ALL) OPTION(*CNTRLD) ENDCCTJOB(*YES) RCDMQMIMG(*YES)
   TIMEOUT( 15 )
   ```
   Where *15* is a timeout value in seconds.

   If the ENDMQM command has not completed within a reasonable period (at least 10 minutes), use the WRKMQM command. This command identifies the queue managers that are still ending. Then force each one in turn to stop by issuing:
   ```
   ENDMQM MQMNAME( QMGRNAME ) OPTION(*IMMED)
   ```
   Where *QMGRNAME* is the name of the queue manager.

   Complete the tidying up of shared memory by issuing the command:
   ```
   ENDMQM MQMNAME(*ALL) OPTION(*IMMED) ENDCCTJOB(*YES) RCDMQMIMG(*NO)
   TIMEOUT( 15 )
   ```
6. If the command in the previous step does not complete, end the subsystem immediately by issuing:
   ```
   ENDSBS SBS(QMQM) OPTION(*IMMED)
   ```
7. If this command also fails, use the operating system command ENDJOB to end all jobs in the subsystem QMQM, as described in the following steps.

   Note: Do not use ENDJOBABN unless you intend to perform an IPL on the machine before starting IBM MQ. Ending IBM MQ jobs using ENDJOBABN can lead to damaged semaphores, which in turn can prevent your queue manager from starting.

   a. If a QMGR must be shut down manually, the recommended order of ending jobs (ENDJOB) is shown in the list that follows. Wait a few minutes for AMQA* or AMQZ* jobs to tidy up.
      1) `RUNMQLSR` - TCP listener (multi-threaded)
      2) `AMQCLMAA` - TCP listener (single-threaded)
      3) `AMQRMPPA` - Channel process pooling job
      4) `RUNMQCHI` - channel initiator
      5) `AMQCRSTA` - receiving MCA jobs
      6) `RUNMQCHL` - sending MCA jobs
      7) `AMQCRS6B` - LU62 receiver channel
      8) `AMQPCSEA` - command server
      9) `RUNMQTRM` - Application trigger monitor
      10) `RUNMQDLQ` - Dead letter queue handler
      11) `AMQFCXBA` - IBM Integration Bus Worker Job
      12) `AMQFQPUB` - Queued Publish/Subscribe Daemon
      13) `RUNMQBRK` - IBM Integration Bus Control Job
      14) `AMQZMUC0` ('0' is a zero) - Utility Manager

15) `AMQZMUF0` ('0' is a zero) - Utility Manager
16) `AMQZMUR0` ('0' is a zero) - Utility Manager
17) `AMQZMGR0` ('0' is a zero) - Process Controller
18) `AMQRRMFA` - cluster repository manager
19) `AMQZDMAA` - deferred message manager
20) `AMQALMPX` - Log Manager
21) `AMQZFUMA` - object authority manager
22) `AMQZLSA0` ('0' is a zero) - LQM agents
23) `AMQZLAA0` ('0' is a zero) - LQM agents
24) `AMQZXMA0` ('0' is a zero) - Execution Controller

b. Issue the following command:
```
ENDMQM MQMNAME( QMGRNAME ) OPTION(*IMMED)
```
c. Issue the following command:
```
ENDMQM MQMNAME(*ALL) OPTION(*CNTRLD) ENDCCTJOB(*YES) RCDMQMIMG(*NO)
TIMEOUT( 05 )
```
Where *05* is a timeout value in seconds.

d. Manually clean up shared memory. Issue the following command:
```
EDTF '/QIBM/UserData/mqm/qmgrs'
```
then:
1) Take option 5 for **&SYSTEM** and check that the following directories are empty: `isem`, `esem`, `msem`, `ssem`, and `shmem`.
2) Take option 5 for **QMGRNAME** and check that the following directories are empty:- `isem`, `esem`, `msem`, `ssem`, and `shmem`.
3) Take option 5 for **&ipcc** in the QMGRNAME directory and check that the following directories are empty:- `isem`, `esem`, `msem`, `ssem`, and `shmem`.
4) Take option 5 for **&qmpersist** in the QMGRNAME directory and check that the following directories are empty:- `isem`, `esem`, `msem`, `ssem`, and `shmem`.
5) Take option 5 for **&app** and check that the following directories are empty: `isem`, `esem`, `msem`, `ssem`, and `shmem`.

8. Load and apply a PTF

Restoring a queue manager to a previous release on IBM i

IBM i

On IBM i, you can restore a queue manager to the previous version of the product from the latest version, if you have made a backup of the system or queue manager. If you have started the queue manager and processed any messages, or changed the configuration, the task cannot give you any guidance on restoring the current state of the queue manager.

Before you begin

1. You must have made a backup of the system or queue manager before you upgraded to the later version. For more information see Backing up and restoring IBM MQ queue manager data
2. If any messages were processed after starting the queue manager, you cannot easily undo the effects of processing the messages. You cannot revert the queue manager to the earlier version of the product in its current state. The task cannot give you any guidance how to deal with subsequent changes that have occurred. For example, messages that were indoubt in a channel, or in a transmission queue on another queue manager, might have been processed. If the queue manager is part of a cluster, then configuration messages and application messages might have been exchanged.

About this task

When you revert to a earlier version of a queue manager, you revert the queue manager to its earlier code level. Queue manager data is reverted to the state it was in when the queue manager was backed up.

Procedure

1. Stop the queue manager.
2. If you performed a slip installation, you must reinstall IBM MQ.
   a. Uninstall the earlier installation.
   b. Reinstall the product from a manufacturing refresh.
   c. Apply the fix pack and interim fixes that restore IBM MQ to its previous level.
   d. Restore the queue manager data from the backup taken before installing the later version.
3. Restart the earlier version queue manager.

Related information:
Backing up and restoring a queue manager

Applying maintenance updates to multi-instance queue managers on IBM i

IBM i

On IBM i, you can use multi-instance queue managers to reduce the outage caused by applying maintenance updates.

Before you begin

Before starting this task, read through the prerequisites described in *Before you begin* in "Applying maintenance level server updates on Windows" on page 573

Before starting this task, see the Maintenance is applied to the IBM MQ installation on a server and not to individual queue managers. Before you apply maintenance, you must stop all the queue managers, and any IBM MQ service, on a server.

If you want a queue manager to keep running while maintenance is applied, you must configure it as a multi-instance queue manager, and have a standby instance running on another server. If the queue manager that you want to keep running is an existing single instance queue manager, you must convert it to a multi-instance queue manager. For prerequisites and guidance how to create a multi-instance queue manager, see Multi-instance queue managers.

You can create a multi-instance queue manager from Version 7.0.1 onwards. If you are running multi-instance queue managers, you then can apply a maintenance update to a running queue manager by switching the active instance to a different server.

Typically, active and standby installations are maintained at the same maintenance level. Consult the maintenance instructions for each update. Consult the instructions to see if it is possible to run the active and standby instances at different maintenance levels. Check whether fail over from higher to lower, or only lower to higher maintenance level is possible.

The instructions for applying a maintenance update might require you to stop a multi-instance queue manager completely.

If you have a primary server for running active queue manager instances, and a secondary server that runs standby instances, you have a choice of updating the primary or secondary server first. If you update the secondary server first, you must switch back to the primary server when both servers have been updated.

If you have active and standby instances on several servers, you must plan in what order you update the servers to minimize the disruption caused by ending the active instances on each server you update.

About this task

Combine the steps in this task with the maintenance update procedure for applying maintenance to an IBM MQ server installation.

Procedure

1. Where the maintenance update procedure instructs you to stop all running queue managers, or quiesce IBM MQ do the following instead: See: “Applying and removing maintenance on IBM i” on page 613.
   a. If the queue manager is running as standby: End the standby by adding the `INSTANCE(*STANDBY)` option to the **ENDMQM** command.
   b. If the queue manager is running as the active instance: End the instance and transfer control to the standby instance by adding the `ALWSWITCH(*YES)` option to the **ENDMQM** command.

      If there is no standby instance running, the command fails, and you must start a standby instance on a different server.
   c. If a queue manager is running as a single instance queue manager, you have no alternative but to stop the queue manager before applying the maintenance update.

   When you complete this step, no queue manager instances are left running on the server you intend to update.
2. Continue with the maintenance update procedure, following the step to issue the **endmqm** command, or quiesce IBM MQ and apply maintenance to the IBM MQ server.
3. When you have completed the maintenance update, restart all the queue managers on the IBM MQ server, permitting standby instances: Add the `STANDBY(*YES)` option to the **STRMQM** command.
4. Repeat the procedure on the standby server, to update its maintenance level.
5. If necessary, switch the active instances back to the primary servers: Use the **ENDMQM** command with the `ALWSWITCH(*YES)` option, and then restart the instances using the **STRMQM** command with the `STANDBY(*YES)` option.

Applying and removing maintenance on z/OS

z/OS

You can install new releases of IBM MQ to update IBM MQ to a new maintenance level.

About this task

LTS From IBM MQ for z/OS Version 9.0, you apply Program Temporary Fixes (PTFs) to the installed code if your enterprise is using the Long Term Support (LTS) release model.

CD If your enterprise is using the Continuous Delivery release (CD release) model, you can select which updates your enterprise requires as each CD release replaces the prior one for that version of IBM MQ.

For more information, see IBM MQ Release Types.

Applying PTFs does not change the version, release, or maintenance level of the code. No queue manager migration is required after applying maintenance. PTFs are grouped into Recommended Service Updates (RSUs) that have been tested together in a Consolidated Service Test (CST); see Consolidated Service Test and the RSU.

On the z/OS LTSR model, maintenance is supplied as Program Temporary Fixes, (PTFs), which are applied and removed using SMP/E. PTFs are specific to a particular set of libraries corresponding to specific release level. Apart from any exceptions documented with the PTFs, PTFs do not change the correct operation of IBM MQ. Nonetheless, you must check that the fixes have not changed the operation of critical programs unexpectedly.

PTFs that apply to a category of software fixes might be grouped together and identified using a fix category. For more information, see IBM Fix category values and descriptions.

When you apply maintenance in the form of PTFs, on z/OS, the impact of the change depends on the extent of the change in VRM level. The VRM codes are explained in "The version naming scheme for IBM MQ for z/OS" on page 569.

PTF updates do not require migration, and are reversible. From Version 7.0.1, all upgrades from Version 6.0 or later are reversible if the **`OPMODE`** has not been set to `NEWFUNC`.

Important: LTS Upgrades to LTS releases only, are reversible.

Related concepts:

"The version naming scheme for IBM MQ for z/OS" on page 569
On IBM MQ for z/OS, releases have a three digit Version, Release, and Modification (VRM) code. To run a queue manager at a different VRM level, you must migrate the queue manager, its applications, and the environment in which it runs. Depending on the migration path, the migration might require more or less effort.

"Queue manager coexistence" on page 664
Queue managers, with different names, can coexist on any server as long as they use the same IBM MQ installation. On z/OS, UNIX, Linux, and Windows, different queue managers can coexist on the same server and be associated with different installations.

Applying upgrades and fixes to IBM MQ

The term upgrade applies to changing the version `V`, release `R`, or modification `M` of a product. The term fix applies to a change in the `F` digit.

About this task

When you upgrade from one release to another, or apply fix packs, or interim fixes, the impact of the change depends on the extent of the change in the `V`, `R`, `M`, `F` level. The `V`, `R`, `M` codes are explained in "The version naming scheme for IBM MQ for Multiplatforms" on page 567.

At each change of `V`, `R`, or `M`, the command level on the queue manager changes, but on a change to `F`, the command level does not.

Multi On IBM MQ for Multiplatforms, after an upgrade has been applied, the only way to *back out* a `V.R.M` change is by:

- Uninstalling the product code and reinstalling the code, or
- Installing the old level of code alongside the existing code and using the setmqm command to associate the queue manager with the other installation.

The general rule, is that if you have carried out an install that causes the command level of the new installation to be updated, and started the queue manager, you cannot *back out* the changes.

Related concepts:

z/OS "Upgrade and migration of IBM MQ on z/OS" on page 785
You can install new releases of IBM MQ to upgrade IBM MQ to a new release, or version level. Multiple installations at the same or different levels can coexist on the same z/OS instance. Running a queue manager at a higher level requires migration.

"New function in maintenance level updates" on page 572
This topic has been removed, as the information contained is not compatible with the Continuous Delivery and Long Term Support release models used in IBM MQ Version 9.0, or later.

Related reference:

z/OS "OPMODE on z/OS" on page 781
The availability of new functions and backward migration for IBM MQ for z/OS is controlled by the **OPMODE** parameter in the **CSQ6SYSP** macro. IBM MQ Version 8.0 new functions that are restricted by **OPMODE** are not available at Version 9.0 unless enabled with **OPMODE**. There are no new functions in Version 9.0 that are restricted by **OPMODE**.

Characteristics of upgrades and fixes

For IBM MQ, the term upgrade applies to changing the version V, release R, or modification M of a product. The term fix applies to a change in the F digit.

Characteristics of fixes

Application of a fix pack, interim fix, or a program temporary fix (PTF) using a maintenance installation tool should be called a fix.

Fixes, applied using a maintenance installation tool, can be rolled back completely, as long as no queue manager migration has taken place on:

- AIX AIX
- Windows Windows
- z/OS z/OS

and IBM MQ is returned to its previous code level.

Attention: z/OS CD On z/OS Continuous Delivery releases, certain PTFs will increase the modification level, and therefore, should be considered an upgrade.

On all other platforms you must reinstall the product.

Characteristics of different types of upgrade

An upgrade can take one of three different forms:

1. Installation of new code on top of existing code. You might be able to roll back an upgrade applied in this way; it depends on the platform. Generally speaking, you cannot roll back the installation of new code. To restore the old code level, you must retain the old installation media, and any fixes you applied.
2. Removal of the old level of code, followed by installation of the new level. The installers on very few platforms require you to remove an old installation first. Needless to say, to restore the old code level, you must reinstall it and any fixes.
3. Side by side installation.

- On z/OS you can install different code levels alongside each other on the same server. In the Job Control Language to start a subsystem, you select the code level to use.
- On UNIX, Linux, and Windows, you associate a queue manager with an installation, and start the queue manager. In IBM MQ, running multiple queue managers at different command levels on the same server is termed queue manager coexistence.

  You must not infer from this, that you can select different installations to run a queue manager at different times. Once a queue manager has been run, it is subject to the rules regarding reverting to earlier or later command levels.

Note: The term upgrade does not imply that an IBM MQ installation can be directly upgraded from one level to another. On some platforms, an upgrade requires that you remove the previous IBM MQ installation. You can retain any queue managers that you have created.

On z/OS, reversibility of an upgrade has two parts; backout of the installation to the previous code level, and reversion of any queue managers that have been started at the new code level, to work with the previous code level again. See "Upgrade and migration of IBM MQ on z/OS" on page 785 for more information.

The rules regarding the reversibility of an queue manager to run on a previous code level is dependent on the platform.

On the following platforms, changes in version, release, or modification level are not fully reversible, but changes in fix level are reversible under certain conditions.
- UNIX
- Linux
- Windows
- IBM i

An irreversible upgrade implies that you must back up the queue managers, or your system, before upgrading, to be able to restore your queue managers. Taking a backup of a queue manager requires you to stop the queue manager. If you do not take a backup, you are not able to restore IBM MQ to its previous level. Any changes you make on the new level cannot be restored onto the backup system. Changes include the creation or deletion of persistent messages, and changes to queue managers, channels, topics, and queues.

Related concepts:

"Upgrade and migration of IBM MQ on z/OS" on page 785
You can install new releases of IBM MQ to upgrade IBM MQ to a new release, or version level. Multiple installations at the same or different levels can coexist on the same z/OS instance. Running a queue manager at a higher level requires migration.

"New function in maintenance level updates" on page 572
This topic has been removed, as the information contained is not compatible with the Continuous Delivery and Long Term Support release models used in IBM MQ Version 9.0, or later.

Related reference:

"OPMODE on z/OS" on page 781
The availability of new functions and backward migration for IBM MQ for z/OS is controlled by the **OPMODE** parameter in the **CSQ6SYSP** macro. IBM MQ Version 8.0 new functions that are restricted by **OPMODE** are not available at Version 9.0 unless enabled with **OPMODE**. There are no new functions in Version 9.0 that are restricted by **OPMODE**.

Upgrading an IBM MQ installation on Windows

To upgrade an IBM MQ server installation on Windows, from one version, release, and modification level to a later one, you can use either the Launchpad or msiexec. To upgrade a client installation, you can use either the GUI installer or msiexec.

About this task

Before you begin, ensure that you have backed up your data.

Important: If you want to apply maintenance instead, for example, from Version 9.0.0.0 to Version 9.0.0, Fix Pack 1, see "Applying and removing maintenance on Windows" on page 573.

Procedure

- To upgrade a server installation, see "Upgrading an IBM MQ server installation using the Launchpad" or "Upgrading an IBM MQ server installation using msiexec" on page 622.
- To upgrade a client installation, see "Upgrading an IBM MQ client installation using the GUI installer" on page 623 or "Upgrading an IBM MQ client installation using msiexec" on page 624.

Upgrading an IBM MQ server installation using the Launchpad

How you upgrade an IBM MQ server installation on Windows to a newer version, release, or modification, using the Launchpad.

Before you begin

Ensure that you have:
1. Stopped all your IBM MQ applications
2. Shut down your listeners
3. Stopped all your queue managers
4. Backed up your data

Important: If you want to apply maintenance instead, for example, from Version 9.0.0.0 to Version 9.0.0, Fix Pack 1, see "Applying and removing maintenance on Windows" on page 573.

Procedure

1. Access the IBM MQ installation image. The location might be the mount point of the DVD, a network location, or a local file system directory.
2. Locate `setup.exe` in the base directory of the IBM MQ installation image.
   - From a DVD, this location might be:
     `E:\setup.exe`
   - From a network location, this location might be:
     `m:\instmqs\setup.exe`
   - From a local file system directory, this location might be:
     `C:\instmqs\setup.exe`
3. Double-click the **Setup** icon to start the installation process. It is possible to run either by:
   - Running `setup.exe` from the command prompt. Or
   - Double-clicking `setup.exe` from Windows Explorer.

If you are installing on a Windows system with UAC enabled, accept the Windows prompt to allow the launchpad to run as elevated. During installation, you might also see Open File - Security Warning dialog boxes that list International Business Machines Limited as the publisher. Click **Run** to allow the installation to continue.

The IBM MQ Installation Launchpad window is displayed.

4. Continue to follow the Launchpad instructions as shown on screen.
5. Select **Installing a new instance**, if you see a panel asking you to choose between installing a new instance, or maintaining or upgrading an existing instance, when you click the **Launch IBM MQ Installer** button. You use the other option when adding or removing features from an already installed IBM MQ.
6. On the next panel, choose between **Install leaving the existing installation(s) untouched** or **Upgrade an existing named installation already on the machine**, and click **Next**.

   Attention: If you do not see this screen, it means that there was no IBM MQ server installation on the machine that could be upgraded by this installer.
7. Follow the installer prompts to upgrade your IBM MQ server installation.

Related tasks:

"Upgrading an IBM MQ server installation using msiexec"
How you upgrade an IBM MQ server installation on Windows to a newer version, release, or modification, using msiexec.

"Upgrading an IBM MQ client installation using the GUI installer" on page 623
How you upgrade an IBM MQ client installation on Windows to a newer version, release, or modification, using the GUI installer.

"Upgrading an IBM MQ client installation using msiexec" on page 624
How you upgrade an IBM MQ client installation on Windows to a newer version, release, or modification, using msiexec.

Upgrading an IBM MQ server installation using msiexec

Windows

How you upgrade an IBM MQ server installation on Windows to a newer version, release, or modification, using msiexec.

Before you begin

Ensure that you have:

1. Stopped all your IBM MQ applications
2. Shut down your listeners
3. Stopped all your queue managers
4. Backed up your data

Important: If you want to apply maintenance instead, for example, from Version 9.0.0.0 to Version 9.0.0, Fix Pack 1, see "Applying and removing maintenance on Windows" on page 573.

Procedure

1. Access the IBM MQ installation image. The location might be the mount point of the DVD, a network location, or a local file system directory.
2. Locate `MSI file` in the MSI directory of the IBM MQ installation image.
   - From a DVD, this location might be:

     E:`\\MSI\IBM MQ.msi`
   - From a network location, this location might be:

     m:\instmqs`\\MSI\IBM MQ.msi`

- From a local file system directory, this location might be:

  `C:\instmqs\\MSI\IBM MQ.msi`

3. Optional: If you are upgrading the only IBM MQ server installation, where the installation has the default value `Installation1` issue the following command:

   ```
   msiexec /i "InstallationImage\MSI\IBM MQ.msi" /q AGREETOLICENSE=YES
   INSTALLATIONNAME="Installation1"
   ```

4. Optional: If you are upgrading an installation on a machine that already has one or more IBM MQ server installations of the level you are upgrading to, you must provide additional parameters to select a free MSI instance ID. See Choosing MSI Instance IDs for multiple server installations for more information.

   In this case, the command might look something like:

   ```
   msiexec /i "Installation Image\MSI\IBM MQ.msi" /q AGREETOLICENSE=YES
   INSTALLATIONNAME="Installation2" NEWINSTANCE=1
   TRANSFORMS=":InstanceId2.mst;1033.mst"
   ```

Related tasks:

"Upgrading an IBM MQ server installation using the Launchpad" on page 621
How you upgrade an IBM MQ server installation on Windows to a newer version, release, or modification, using the Launchpad.

"Upgrading an IBM MQ client installation using the GUI installer"
How you upgrade an IBM MQ client installation on Windows to a newer version, release, or modification, using the GUI installer.

"Upgrading an IBM MQ client installation using msiexec" on page 624
How you upgrade an IBM MQ client installation on Windows to a newer version, release, or modification, using msiexec.

Upgrading an IBM MQ client installation using the GUI installer

Windows

How you upgrade an IBM MQ client installation on Windows to a newer version, release, or modification, using the GUI installer.

Before you begin

Ensure that you have:

1. Stopped all your IBM MQ applications
2. Shut down your listeners
3. Stopped all your queue managers
4. Backed up your data

Procedure

1. Access the IBM MQ installation image. The location might be the mount point of the DVD, a network location, or a local file system directory.
2. Locate `setup.exe` in the base directory of the IBM MQ installation image.
   - From a DVD, this location might be:

     `E:\setup.exe`
   - From a network location, this location might be:

     `m:\instmqs\setup.exe`
   - From a local file system directory, this location might be:

     `C:\instmqs\setup.exe`
3. Double-click the **Setup** icon to start the installation process. It is possible to run either by:
   - Running `setup.exe` from the command prompt. Or

- Double-clicking `setup.exe` from Windows Explorer.

If you are installing on a Windows system with UAC enabled, accept the Windows prompt to allow the launchpad to run as elevated. During installation, you might also see Open File - Security Warning dialog boxes that list International Business Machines Limited as the publisher. Click **Run** to allow the installation to continue.

The IBM MQ Installation Launchpad window is displayed.

4. Continue to follow the Launchpad instructions as shown on screen.
5. Select **Installing a new instance**, if you see a panel asking you to choose between installing a new instance, or maintaining or upgrading an existing instance, when you click the **Launch IBM MQ Installer** button. You use the other option when adding or removing features from an already installed IBM MQ.
6. On the next panel, choose between **Install leaving the existing installation(s) untouched** or **Upgrade an existing named installation already on the machine**, and click **Next**.

   Attention: If you do not see this screen, it means that there was no IBM MQ client installation on the machine that could be upgraded by this installer.
7. Follow the installer prompts to upgrade your IBM MQ client installation.

Related tasks:

"Upgrading an IBM MQ client installation using msiexec"
How you upgrade an IBM MQ client installation on Windows to a newer version, release, or modification, using msiexec.

"Upgrading an IBM MQ server installation using the Launchpad" on page 621
How you upgrade an IBM MQ server installation on Windows to a newer version, release, or modification, using the Launchpad.

"Upgrading an IBM MQ server installation using msiexec" on page 622
How you upgrade an IBM MQ server installation on Windows to a newer version, release, or modification, using msiexec.

Upgrading an IBM MQ client installation using msiexec

Windows

How you upgrade an IBM MQ client installation on Windows to a newer version, release, or modification, using msiexec.

Before you begin

Ensure that you have:

1. Stopped all your IBM MQ applications
2. Shut down your listeners
3. Stopped all your queue managers
4. Backed up your data

Procedure

1. Access the IBM MQ installation image. The location might be the mount point of the DVD, a network location, or a local file system directory.
2. Locate `MSI file` in the MSI directory of the IBM MQ installation image.
   - From a DVD, this location might be:

     `*E:*\\MSI\IBM MQ.msi`
   - From a network location, this location might be:

     `*m:\instmqs*\\MSI\IBM MQ.msi`
   - From a local file system directory, this location might be:

```
C:\instmqs\\MSI\IBM MQ.msi
```

3. Optional: If you are upgrading the only IBM MQ client installation, where the installation has the default value `Installation1` issue the following command:

   ```
   msiexec /i "InstallationImage\MSI\IBM MQ.msi" /q AGREETOLICENSE=YES
   INSTALLATIONNAME="Installation1"
   ```

4. Optional: If you are upgrading an installation on a machine that already has one or more IBM MQ client installations of the level you are upgrading to, you must provide additional parameters to select a free MSI instance ID. See Choosing MSI Instance IDs for multiple client installations for more information.

   In this case, the command might look something like:

   ```
   msiexec /i "Installation Image\MSI\IBM MQ.msi" /q AGREETOLICENSE=YES
   INSTALLATIONNAME="Installation2" NEWINSTANCE=1
   TRANSFORMS=":InstanceId2.mst;1033.mst"
   ```

Related tasks:

"Upgrading an IBM MQ client installation using the GUI installer" on page 623
How you upgrade an IBM MQ client installation on Windows to a newer version, release, or modification, using the GUI installer.

"Upgrading an IBM MQ server installation using the Launchpad" on page 621
How you upgrade an IBM MQ server installation on Windows to a newer version, release, or modification, using the Launchpad.

"Upgrading an IBM MQ server installation using msiexec" on page 622
How you upgrade an IBM MQ server installation on Windows to a newer version, release, or modification, using msiexec.

Migrating IBM MQ

Migration is the conversion of programs and data to work with a new code level of IBM MQ. Some types of migration are required, and some are optional. Queue manager migration is never required after applying a maintenance level update, that does not change the command level. Some types of migration are automatic, and some are manual. Queue manager migration is typically automatic and required after releases and manual and optional after a maintenance level upgrade that introduces a new function. Application migration is typically manual and optional.

Before you begin

Before upgrading your IBM MQ installation or migrating your queue managers, you must read "Changes that affect migration" on page 626 to identify what migration tasks you must plan for.

About this task

Whenever you upgrade IBM MQ to a new release that changes its command level, migration is performed by the queue manager. Whenever you upgrade IBM MQ to a new maintenance or fix level, which introduces a new function using a new command level, you can migrate the queue manager to use the new command level and thereby the new function.

If you start a queue manager running on a later release level, then migration of the queue manager to that release level is required. The migration tasks you must perform to migrate from one release to another are documented in "Migrating a queue manager on Windows" on page 691; see also "Changes that affect migration" on page 626.

Multi On IBM MQ for Multiplatforms, you cannot easily revert to a previous level of IBM MQ after installation. If you install a copy of IBM MQ obtained from Passport Advantage or from physical media, the installer uninstalls IBM MQ, if it is present. It then installs the new level of IBM MQ. To revert to the previous level of IBM MQ, you must keep the earlier installation image and any fixes you applied.

Then you must uninstall the new level, reinstall the previous release level, and reapply the required fixes. If you have started any queue managers at the later level, they will not work with the restored level of IBM MQ[1] . To restore IBM MQ to its previous level, after starting any queue managers, you must first back up the queue managers. You can then restore the backup queue managers after restoring the previous level of IBM MQ.

z/OS LTS On IBM MQ for z/OS, while **OPMODE** is set to COMPAT, it is possible to backwards migrate from a Long Term Support (LTS) release. For more information, see "Backward migration to earlier supported releases of IBM MQ for z/OS" on page 789.

z/OS CD Backwards migration is not supported for a Continuous Delivery (CD) release on z/OS.

Related concepts:

"The version naming scheme for IBM MQ for Multiplatforms" on page 567
From IBM MQ Version 9.0, releases have a three digit Version, Release, and Modification (VRM) code or a four-digit Version, Release, Maintenance, and Fix (VRMF) level code.

"Multi-installation queue manager coexistence on UNIX, Linux, and Windows" on page 668
You can install multiple copies of IBM MQ for UNIX, Linux, and Windows on the same server. The installations must be at Version 7.1 or later, with one exception. One Version 7.0.1 installation, at fix pack level 6, or later, can coexist with multiple Version 7.1, or later installations.

"Queue manager coexistence" on page 664
Queue managers, with different names, can coexist on any server as long as they use the same IBM MQ installation. On z/OS, UNIX, Linux, and Windows, different queue managers can coexist on the same server and be associated with different installations.

Related information:

Backing up and restoring a queue manager

Changes that affect migration

Changes to the product might affect the migration of a queue manager from an earlier release to the current release of IBM MQ, or affect existing applications or configurations. Review these changes before upgrading queue managers to the latest product version and decide whether you must plan to make changes to existing applications, scripts, and procedures before starting to migrate your systems.

Changes in the current release

For details of changes in the current release, including those that affect migration, see the following information:

- V 9.0.0 What's new and changed in IBM MQ Version 9.0
- CD What's new and changed in Version 9.0.x Continuous Delivery
- LTS What's changed in Version 9.0.0.x Long Term Support
- LTS CD Readme for IBM MQ V9.0 and its maintenance

Changes in earlier releases

For information about what changed in an earlier release of the product, see the *What's changed* section in the product documentation for that release:

- What's changed in IBM MQ Version 8.0

1. Unless you installed a later maintenance level upgrade, not a new release or version: then you could revert to an earlier maintenance level by reinstalling the earlier maintenance level upgrade. Queue manager data is compatible between maintenance levels.

- What's changed in IBM WebSphere MQ Version 7.5
- What's changed in IBM WebSphere MQ Version 7.1
- What's changed in IBM WebSphere MQ Version 7.0.1

Restrictions on reversing queue manager migration

Attention:

Multi On IBM MQ for Multiplatforms, you cannot reverse queue manager migration to remove the effect of changes. This restriction applies whether your enterprise uses the Long Term Support (LTS) release or Continuous Delivery (CD) release model.

z/OS LTS On IBM MQ for z/OS, you can reverse queue manager migration as long as you have not enabled new function, and are using the LTS release model. You enable new function, for a queue manager on the LTS release model, by setting the **OPMODE** parameter to (NEWFUNC, 900).

For further information, see IBM MQ release types.

Related concepts:

"Migration concepts and methods" on page 645
An overview of the various concepts and methods for migrating from one release of the product to another.

"Migration considerations for Version 8.0 or later on Windows" on page 686
From Version 8.0, a number of changes were made for IBM MQ for Windows. You must understand these changes before planning any migration tasks for Version 8.0 or later on Windows.

"Program and data directory locations on Windows" on page 688
The installation location for IBM MQ program binary and data files on Windows depends on the IBM MQ version you are installing and whether this is the first time IBM MQ is being installed.

What changed in IBM MQ Version 8.0

Changes to functions and resources, including stabilizations, deprecations and removals, that occurred in IBM MQ Version 8.0 are summarized in this section, with links to further information.

This information has moved. See What changed in IBM MQ Version 8.0.

APIs: change to the operation of API exits:

API exits which themselves make API calls, rather than using the **MQIEP** structure, should be linked with an IBM MQ API library, otherwise they will fail to load inside **runmqsc**.

This information has moved. See APIs: Change to the operation of API exits.

Change to STOPPED status on a SVRCONN channel:

From IBM MQ Version 8.0 STOPPED channel status is allowed to persist a queue manager restart. This behavior applies to all platforms excluding z/OS.

This information has moved. See Change to STOPPED status on a SVRCONN channel.

Command changes:

From IBM MQ Version 8.0, there is a change of behavior in a number of commands.

This information has moved. See:

- Command changes: crtmqenv command

- Command changes: dmpmqcfg output
- Command changes: results displayed in response to DISPLAY CONN command from runmqsc
- Changes to the runmqsc command: new client modes for MQSC
- Command changes: removal of migmbbrk and CSQUMGMB

Command level changes at Version 8.0:

The command level on all platforms changed to 800 in IBM MQ Version 8.0. On certain platforms it changed to 801 in IBM MQ Version 8.0.0, Fix Pack 2, and to 802 in IBM MQ Version 8.0.0, Fix Pack 3. The command level remained at 802 for those platforms in IBM MQ Version 8.0.0, Fix Pack 4.

This information has moved. See Command level: changes.

Deprecations:

A number of features are deprecated from IBM MQ Version 8.0.

This information has moved. See:

- Solaris Deprecation: linking with libmqmcs and libmqmzse libraries
- Deprecation: Managed File Transfer Web Gateway
- Deprecation: SSLv3 protocol
- Deprecation: weaker cryptographic algorithms
- Deprecation: IBM MQ Explorer Service Definition Wizard
- Deprecation: IBM MQ transport for SOAP client and HTTP bridge

Extended start events for multi-instance queue managers:

From IBM MQ Version 8.0.0, Fix Pack 3, start events for multi-instance queue managers are extended to allow system monitoring applications to see when a multi-instance queue manager has failed over and where it is now running.

This information has moved. See Extended start events for multi-instance queue managers.

Additions to supported features on IBM i: IBM i

From IBM MQ Version 8.0, IBM i supports split cluster transmission queues and Advanced Message Security.

This information has moved. See IBM i: changes for IBM MQ Version 8.0.

Java and JMS: changes to character string conversion:

From IBM MQ Version 8.0, some of the default behavior regarding character string conversion with Java and JMS clients has changed.

This information has moved. See Java and JMS: changes to character string conversion.

Java and JMS: changes to CipherSuite support:

From IBM MQ Version 8.0 the support for SSL/TLS CipherSuites in the IBM MQ classes for Java and IBM MQ classes for JMS has changed.

This information has moved. See Java and JMS: changes to CipherSuite support.

Java and JMS: resource adapter changes:

From IBM MQ Version 8.0.0, Fix Pack 6, the IBM MQ resource adapter IVT application is updated to support WildFly V10.

This information has moved. See Java and JMS: resource adapter changes.

Java: changes for Java 7:

In order to support Java 7, changes have been made to IBM MQ classes for Java and IBM MQ classes for JMS.

This information has moved. See Java: changes for Java 7.

Java: Changes to IBM MQ classes for Java:

From IBM MQ Version 8.0, an unwanted dependency on `connector.jar` has been removed and changes have been made to APIs for public classes.

This information has moved. See Java: changes to IBM MQ classes for Java.

Java: changes to Java tracing behavior:

From IBM MQ Version 8.0 the ways to control trace for Java-language clients has been updated.

This information has moved. See Java: changes to Java tracing behavior.

Java: changes to MQException and MQDataException class logging:

From IBM MQ Version 8.0, the logging functionality of the com.ibm.mq.MQException class, which wrote details of new MQExceptions to a log, has been removed.

This information has moved. See Java: changes to MQException and MQDataException class logging.

JMS: changes to Bytes and Stream message states:

From IBM MQ Version 8.0.0, Fix Pack 2, a property can be set to define whether IBM MQ classes for JMS should set the state of a message that has just been sent to read only, or write only.

This information has moved. See JMS: changes to Bytes and Stream message states.

JMS: Exception listener changes in Version 8.0:

In IBM MQ Version 8.0, when using IBM MQ classes for JMS, changes are made to the way in which an application's JMS ExceptionListener is invoked.

This information has moved. See JMS: Exception listener changes in Version 8.0.

JMS: changes to IBM MQ classes for JMS:

From IBM MQ Version 8.0, a number of changes are made to IBM MQ classes for JMS.

This information has moved. See JMS: changes to IBM MQ classes for JMS.

JMS: changes to JAR files for JMS 2.0:

From IBM MQ Version 8.0, the `jms.jar` file is a JMS 2.0 set of interfaces.

This information has moved. See JMS: changes to JAR files for JMS 2.0.

JMS: changes to `PROVIDERVERSION` property:

The JMS **`PROVIDERVERSION`** property selects whether a Java application publishes and subscribes using the queued command message interface, or the integrated call interface. For IBM MQ Version 8.0 in addition to normal mode and migration mode, you can also select normal mode with restrictions.

This information has moved. See JMS: changes to PROVIDERVERSION property.

Managed File Transfer: changes for Version 8.0.0, Fix Pack 6:

For IBM MQ Version 8.0.0, Fix Pack 6, a number of changes are made to Managed File Transfer.

This information has moved. See Managed File Transfer: changes for Version 8.0.0, Fix Pack 6.

Changes to cluster topic behavior:

From IBM MQ Version 8.0, the behavior has changed when the publication (**`PUB`**) parameter differs across clustered topic definitions of the same name.

This information has moved. See Changes to cluster topic behavior.

Publish/subscribe: changes to DISPLAY TOPIC and DISPLAY TPSTATUS output:

From IBM MQ Version 8.0 the output of the **`DISPLAY TOPIC`** command now includes the parameters **`CLROUTE`** and **`CLSTATE`** and the **`DISPLAY TPSTATUS`** command now includes the parameter **`CLROUTE`**.

This information has moved. See Publish/subscribe: changes to DISPLAY TOPIC and DISPLAY TPSTATUS output.

Publish/subscribe: changes to how the subscription selection string is evaluated:

From IBM MQ Version 8.0, message selection for publish/subscribe messaging is made on the message as sent by the publisher, rather than on the message as eventually received by the subscriber.

This information has moved. See Publish/subscribe: changes to how the subscription selection string is evaluated.

Publish/subscribe: changes to proxy subscription propagation using PROXYSUB(FORCE):

From IBM MQ Version 8.0, when you configure a topic object with the parameter **`PROXYSUB`** set to `FORCE`, IBM MQ automatically disables individual proxy subscriptions for topic strings below this point in the topic tree.

This information has moved. See Publish/subscribe: changes to proxy subscription propagation using PROXYSUB(FORCE).

Queue managers: change to default TCP buffer size:

From IBM MQ Version 8.0, there is a change to the TCP stanza values for send and receive buffer sizes for new queue managers, which without manual tuning, previously defaulted to a fixed size 32Kb buffer. z/OS This change does not apply to z/OS.

This information has moved. See Queue managers: change to default TCP buffer size.

Queue managers: Version attribute added for queue managers in a cluster:

From IBM MQ Version 8.0 you can display the version of IBM MQ with which the cluster queue manager is associated in the form of `VVRRMMFF` where `VV` is the version, `RR` is the release, `MM` is the maintenance level, and `FF` is the fix level.

This information has moved. See Queue managers: Version attribute added for queue managers in a cluster.

Security: certificate revocation checking:

From IBM MQ Version 8.0, you can determine how certificate revocation checking is configured if the client connect call uses an SSL/TLS channel by using the **`ClientRevocationChecks`** attribute in the SSL stanza of the client configuration file.

This information has moved. See Security: certificate revocation checking.

Security: changes to CipherSuite support:

From IBM MQ Version 8.0, the SHA-2 support that is already provided in earlier releases is extended.

This information has moved. See Security: changes to CipherSuite support.

Security: Deprecated CipherSpecs:

From IBM MQ Version 8.0.0, Fix Pack 6, the following CipherSpecs are deprecated:

This information has moved. See Security: deprecated CipherSpecs.

Security: GSKit version updated:

The GSKit version has been updated in IBM MQ Version 8.0.0, Fix Pack 6. The new version of GSKit alters the stash file format that is used when you generate an `.sth` file to stash the key database password. Stash files that are generated with this version of GSKit are not readable by earlier versions of GSKit.

This information has moved. See What's new and changed in Version 8.0.0, Fix Pack 6.

Security: new CONNAUTH CHCKLOCL parameter:

From IBM MQ Version 8.0, a new queue manager security parameter, **`CONNAUTH CHCKLOCL`**, is introduced.

This information has moved. See Security: new CONNAUTH CHCKLOCL parameter.

Security: OAM user-based permissions on UNIX and Linux systems:

From IBM MQ Version 8.0, on UNIX and Linux systems, the object authority manager (OAM) can now use user-based authorization as well as group-based authorization.

This information has moved. See Security: OAM user-based permissions on UNIX and Linux systems.

Security: Disable AMS at the client:

You can disable Advanced Message Security at the client, to prevent errors when they are connecting to queue managers that are running on IBM WebSphere MQ Version 7.1 or earlier. From IBM MQ Version 8.0, the environment variable that is used to disable AMS for C clients, is changed.

This information has moved. See Security: Disable IBM MQ AMS at the client.

Security: password protection on client applications:

From IBM MQ Version 8.0, any passwords that you send using the MQCSP structure can be either protected, by using IBM MQ functionality, or encrypted, by using SSL/TLS encryption.

This information has moved. See Security: password protection on client applications.

Security: reduction in channel send exit buffer space for SSL and TLS:

From IBM MQ Version 8.0, the maximum amount of space in the transmission buffer that an SSL or TLS channel send exit can reserve by the `MQCXP ExitSpace` field has been reduced by 924 bytes to 14,328 bytes.

This information has moved. See Security: reduction in channel send exit buffer space for SSL and TLS.

Solaris client: change in method of installation:

From IBM MQ Version 8.0 the *.img* file has been removed. The client is delivered in an unpacked form.

This information has moved. See Solaris client: change in method of installation.

Telemetry: changes to MQ Telemetry Clients Software Development Kit (SDK):

From IBM MQ Version 8.0 the MQ Telemetry SDK is no longer supplied as part of the product.

This information has moved. See Telemetry: changes to MQ Telemetry Clients Software Development Kit (SDK).

changes from IBM MQ Version 8.0 on Windows: Windows

From IBM MQ Version 8.0, changes have been made to the default installation location for the server and 64-bit client installations. There has also been a change in the default data path for all installations. The compiler that is used from IBM MQ Version 8.0 on Windows has been changed.

This information has moved. See Windows: changes for IBM MQ Version 8.0.

Change to display of mapped IP addresses in error message inserts on z/OS: z/OS

From IBM MQ Version 8.0, mapped IP addresses shown in error messages in the CHINIT address space on z/OS are shown without the mapped prefix.

This information has moved. See z/OS: change to display of mapped IP addresses in error message inserts.

Change to MAKECLNT on z/OS: z/OS

From IBM MQ Version 8.0, the MAKECLNT CSQUTIL tool is stabilized. You should use **runmqsc -n** instead.

This information has moved. See z/OS: change to MAKECLNT.

Changes to log RBA and URID lengths on z/OS: z/OS

From IBM MQ Version 8.0, log RBAs and Unit of Recovery Identifiers (URID) in console messages and command responses are 8 bytes long, displayed as 16 character hexadecimal values. This is the case regardless of whether Version 8.0 new functions have been enabled with OPMODE, and whether the queue manager has been migrated to use 8 byte log RBAs.

This information has moved. See z/OS: changes to log RBA and URID lengths.

Changes to Managed File Transfer on z/OS: z/OS

Changes have been made between WebSphere MQ File Transfer Edition on z/OS and Managed File Transfer on IBM MQ for z/OS.

This information has moved. See z/OS: changes to Managed File Transfer.

z/OS: channel initiator SMF records: z/OS

The channel initiator (CHINIT) can produce SMF statistics records and accounting records with information on tasks and channels.

This information has moved. See z/OS: channel initiator SMF records.

WLM/DNS no longer supported on z/OS: z/OS

WLM/DNS is no longer supported by the z/OS Communications Server, so the queue manager attributes DNSWLM and DNSGROUP are no longer used from IBM MQ for z/OS Version 8.0.

This information has moved. See z/OS: WLM/DNS no longer supported.

What changed in IBM WebSphere MQ Version 7.0.1, Version 7.1 and Version 7.5

Changes that affect the migration of a queue manager from Version 7.0.1, Version 7.1 and Version 7.5 to the latest release are listed.

This information has moved. See:
- What's changed in IBM WebSphere MQ Version 7.5
- What's changed in IBM WebSphere MQ Version 7.1
- What's changed in IBM WebSphere MQ Version 7.0.1

Version 7.0.1 features and their supported APIs:

Use this information to learn about the IBM WebSphere MQ Version 7.0.1 APIs with features and environments that might not be as fully supported as the C MQI.

This information has moved. See New features and their supported APIs for Version 7.0.1.

Version 7.0.1 features and their supported APIs: JMS:

Use this information to learn about the IBM WebSphere MQ Version 7.0.1 APIs with features and environments that might not be as fully supported as the C MQI.

This information has moved. See New features and their supported APIs for Version 7.0.1.

Version 7.0.1 features and their supported APIs: Java:

Use this information to learn about the IBM WebSphere MQ Version 7.0.1 APIs with features and environments that might not be as fully supported as the C MQI.

This information has moved. See New features and their supported APIs for Version 7.0.1.

Version 7.0.1 features and their supported APIs: XMS .NET:

Use this information to learn about the IBM WebSphere MQ Version 7.0.1 APIs with features and environments that might not be as fully supported as the C MQI.

This information has moved. See New features and their supported APIs for Version 7.0.1.

Version 7.0.1 features and their supported APIs: XMS C and C++:

Use this information to learn about the IBM WebSphere MQ Version 7.0.1 APIs with features and environments that might not be as fully supported as the C MQI.

This information has moved. See New features and their supported APIs for Version 7.0.1.

Version 7.0.1 features and their supported APIs: .NET:

Use this information to learn about the IBM WebSphere MQ Version 7.0.1 APIs with features and environments that might not be as fully supported as the C MQI.

This information has moved. See New features and their supported APIs for Version 7.0.1.

Channel authentication:

From IBM WebSphere MQ Version 7.1 onwards, when you migrate a queue manager from an earlier release, channel authentication using channel authentication records is disabled. Channels continue to work as before. If you create a queue manager in Version 9.0, channel authentication using channel authentication records is enabled, but with minimal additional checking. Some channels might fail to start.

This information has moved. See Channel authentication.

Change in behavior of the `endmqm` command:

From IBM WebSphere MQ Version 7.5 onwards, issuing an **`endmqm`** command and **`dspmq`** command immediately after each other might return misleading status.

This information has moved. See Change in behavior of the endmqm command.

Changes to cluster error recovery (on servers other than z/OS): Multi

From IBM WebSphere MQ Version 7.1 onwards, 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.

This information has moved. See Changes to cluster error recovery (on servers other than z/OS).

Change in behavior of MQS_REPORT_NOAUTH:

In IBM WebSphere MQ Version 7.1, the default behavior of MQS_REPORT_NOAUTH was changed to *TRUE*.

This information has moved. See Change in behavior of MQS_REPORT_NOAUTH.

Connect to multiple queue managers and use `MQCNO_FASTPATH_BINDING`:

From Version 7.1 onwards, only the first connection can be set to `MQCNO_FASTPATH_BINDING`. Applications that connect to queue managers using the `MQCNO_FASTPATH_BINDING` binding option might fail with an error and reason code `MQRC_FASTPATH_NOT_AVAILABLE` if you do not follow the rules for using fast path connections.

This information has moved. See Connect to multiple queue managers and use `MQCNO_FASTPATH_BINDING`.

Custom scripts:

In IBM MQ for Windows, custom scripts that have been used in an earlier release to install packages might fail if any packages have been renamed, removed, or added in the later release to which you are migrating.

This information has moved. See Custom scripts.

Changes to data types:

A number of data types have changed between IBM WebSphere MQ Version 7.0.1 and later releases, and new data types have been added.

This information has moved. See
- Changes to data types (Version 7.1)
- Changes to data types (Version 7.5)

Default transmission queue restriction:

The product documentation in versions of IBM WebSphere MQ before Version 7.1 warned about defining the default transmission queue as `SYSTEM.CLUSTER.TRANSMIT.QUEUE`, but no error was reported. In Version 7.1, any attempt to set or use a default transmission queue that is defined as `SYSTEM.CLUSTER.TRANSMIT.QUEUE` results in an error.

This information has moved. See Default transmission queue restriction.

Display channel and cluster status: Switching:

From IBM WebSphere MQ Version 7.5 onwards, a cluster-sender channel that is switching its configuration to a different cluster transmission queue has a new channel state: Switching.

This topic has moved. See Display channel and cluster status: Switching.

Changes to `dspmqver` output:

From IBM WebSphere MQ Version 7.1 onwards, new types of information are displayed by **`dspmqver`** to support multiple installations. The changes might affect existing administrative scripts that you have written to manage a release of IBM WebSphere MQ before IBM WebSphere MQ Version 7.1.

This information has moved. See Changes to dspmqver output.

For information about the **`dspmqver`** command, see dspmqver.

Exits and installable services: Multi

When migrating to IBM WebSphere MQ Version 7.1 or later on Multiplatforms, if you install the product in a non-default location, you must update your exits and installable services. Data conversion exits generated using the **`crtmqcvx`** command must be regenerated using the updated command.

This information has moved. See Exits and installable services.

Fewer IBM MQ MQI client log messages:

Before Version 7.1, an IBM WebSphere MQ MQI client used to report every failed attempt to connect to a queue manager when processing a connection name list. From Version 7.1, only if the failure occurs with the last connection in the list is a message written to the queue manager error log.

This information has moved. See Fewer IBM WebSphere MQ MQI client log messages.

Java: Change in behavior of default value of MQEnvironment.userID:

From IBM WebSphere MQ Version 7.1, a change to the behavior of the default value of MQEnvironment.userID results in an exception if an application attempts to connect to a queue manager using the CLIENT transport through a channel that does not have a security exit defined.

This information has moved. See Java: Change in behavior of default value of MQEnvironment.userID.

Java: Different message property data type returned:

From IBM WebSphere MQ Version 7.1, if the data type of a message property is set, the same data type is returned when the message is received. This is different from IBM WebSphere MQ Version 7.0.1, where in some circumstances properties set with a specific type were returned with the default type String.

This information has moved. See Java: Different message property data type returned.

JMS: Change in behavior of the default user identifier value:

From IBM WebSphere MQ Version 7.1, a change to the behavior of the default user identifier value results in an exception if an application attempts to connect to a queue manager using the CLIENT transport through a channel that does not have a security exit defined.

This information has moved. See JMS: Change in behavior of the default user identifier value.

JMS: Exception listener changes:

In later releases of the product, JMS exception listeners behave differently than they did before Version 7.0. Applications might receive more or fewer exceptions than they did in earlier releases.

This information has moved. See:

- JMS: Exception listener changes in Version 7.0.
- .

JMS: some objects are no longer serializable:

JMS objects such as JMS Connections and JMS Sessions, which used to be serializable when using the IBM WebSphere MQ classes for JMS for IBM WebSphere MQ Version 7.0.1, are no longer serializable when using IBM WebSphere MQ Version 7.1 or later.

This information has moved. See JMS: some objects are no longer serializable .

JMS: Reason code changes:

From IBM WebSphere MQ Version 7.1, some reason codes returned in JMS exceptions have changed. The changes affect `MQRC_Q_MGR_NOT_AVAILABLE` and `MQRC_SSL_INITIALIZATION_ERROR`.

This information has moved. See JMS: reason code changes.

JMS: ResourceAdapter object configuration:

When WebSphere Application Server connects to IBM MQ it creates message driven beans (MDBs) using JMS connections. From Version 7.1, these MDBs can no longer share one JMS connection. The configuration of ResourceAdapter object is migrated so that there is a single MDB for each JMS connection.

This information has changed. See JMS: ResourceAdapter object configuration.

IBM MQ Explorer changes:

From Version 7.1, IBM WebSphere Eclipse Platform is not required to run IBM MQ Explorer and is therefore no longer shipped with the product. The change makes no difference to administrators who run IBM MQ Explorer. For developers who run IBM MQ Explorer in an Eclipse development environment, a change is necessary. You must install and configure a separate Eclipse environment to be able to switch between IBM MQ Explorer and other perspectives.

This information has moved. See IBM WebSphere MQ Explorer changes.

MQI and PCF reason code changes:

Some reason codes that affect some existing programs changed in IBM WebSphere MQ Version 7.1.

This information has moved. See MQI and PCF reason code changes.

Publish/Subscribe: Delete temporary dynamic queue:

If a subscription is associated with a temporary dynamic queue, when the queue is deleted, the subscription is deleted. Publish/subscribe applications migrated from WebSphere Message Broker are unchanged. The change does not affect the behavior of integrated publish/subscribe applications, which are written using the MQI publish/subscribe interface.

This information has moved. See Publish/Subscribe: Delete temporary dynamic queue.

Queue manager logs: Default sizes increased:

From IBM WebSphere MQ Version 7.1, the default size of a queue manager log files changed to 4096. The queue manager error log increased from 256 KB to 2 MB on some platforms. The change affects both new and migrated queue managers.

This information has moved. See Queue manager log files and error logs: Default sizes increased.

Removal of `dspmqsver` command:

Before IBM WebSphere MQ Version 7.5, the **`dspmqsver`** command was used to display the version of Advanced Message Security.

This information has moved. See Removal of dspmqsver command.

Security: `SSLPEER` and `SSLCERTI` changes:

IBM WebSphere MQ Version 7.1 or later obtains the Distinguished Encoding Rules (DER) encoding of the certificate and uses it to determine the subject and issuer distinguished names. The subject and issuer distinguished names are used in the `SSLPEER` and `SSLCERTI` fields. A `SERIALNUMBER` attribute is also included in the subject distinguished name and contains the serial number for the certificate of the remote partner. Some attributes of subject and issuer distinguished names are returned in a different sequence from releases before Version 7.1.

This information has moved. See SSLPEER and SSLCERTI changes.

Security: Disable AMS at the client:

From Version 7.5, you can disable Advanced Message Security at the client, for Java and C clients, to prevent errors when they are connecting to queue managers that are running on earlier versions of the product.

This information has moved. See What's changed in IBM WebSphere MQ Version 7.5 Fix Packs.

Telemetry: Installer integrated with IBM MQ:

From Version 7.1, MQ Telemetry is no longer installed separately from IBM MQ. It is installed as a component of the main product. If you installed MQ Telemetry with Version 7.0.1, you must uninstall it before installing Version 7.1 or later.

This information has moved. See Telemetry: Installer integrated with IBM WebSphere MQ.

Telemetry: Support for the MQTT protocol over WebSockets:

IBM WebSphere MQ Version 7.5.0, Fix Pack 1 and later supports the MQTT protocol over WebSockets. This enables it to be a server for clients using the MQTT messaging client for JavaScript.

This information has moved. See What's changed in IBM WebSphere MQ Version 7.5 Fix Packs.

Shared objects on AIX: AIX

On AIX, for Version 7.1. and later, the `.a` shared objects in the `lib64` directory contains both the 32 bit and 64 bit objects. A symlink to the `.a` file is also placed in the `lib` directory. The AIX loader can then correctly pick up the correct object for the type of application being run.

This means that IBM MQ applications can run with the `LIBPATH` containing either the `lib` or `lib64` directory, or both.

This information has moved. See AIX: Shared objects.

`/usr/lpp/mqm` symbolic link removed on AIX: AIX

Before Version 6.0, IBM WebSphere MQ placed a symbolic link in `/usr/lpp/mqm` on AIX. The link ensured queue managers and applications migrated from IBM WebSphere MQ versions before Version 5.3 continued to work, without change. However, this link is not created in Version 7.1, or later.

This information has moved. See AIX: /usr/lpp/mqm symbolic link removed.

Building applications for TXSeries on AIX, HP-UX, and Solaris:

From Version 7.1, applications that link to TXSeries® must load the `mqzi_r` library, not the `mqz_r` library. These applications must load `mqzi_r` from the installation that is associated with the queue manager to which the application is connected.

This information has moved. See AIX, HP-UX, and Solaris: Building applications for TXSeries.

GSKit: Recompile LinuxC++ applications and update run time libraries: Multi

C++ IBM MQ MQI client and server applications on Linux must be recompiled using a supported version of the GNU Compiler Collection (GCC). The C++ run time libraries for this version of GCC must be installed in `/usr/lib` or `/usr/lib64`.

This information has moved. See GSKit: Recompile Linux C++ applications and update run time libraries.

Increased shared memory allocation required on Linux: Linux

The maximum amount of shared memory (`SHMMAX`) to allocate on Linux systems. The default system allocation is 32 MB. IBM MQ starts by allocating 64 MB and increases its allocation on demand by doubling its previous allocation. On a production system set `SHMMAX` to at least 256 MB to accommodate additional allocations.

This information has moved. See Linux: Increased shared memory allocation required.

UNIX and Linux : `crtmqlnk` and `dltmqlnk` removed:

Before Version 7.1 , the **`crtmqlnk`** and **`dltmqlnk`** commands created symbolic links in subdirectories of `/usr`. From Version 7.1 onwards, you must use the **`setmqinst`** command instead.

This information has moved. See UNIX and Linux: crtmqlnk and dltmqlnk removed.

UNIX and Linux: Message catalogs moved:

From Version 7.1, message catalogs are no longer stored in the system directories. To support multiple installations, copies of the message catalogs are stored with each installation. If you are migrating from a release before Version 7.1 and you want messages only in the locale of your system, the change has no affect on your system. If you have customized the way in which the search procedure selects a message catalog, then the customization might no longer work correctly.

This information has moved. See UNIX and Linux: Message catalogs moved.

UNIX and Linux: MQ services and triggered applications:

From Version 7.1, both `LD_LIBRARY_PATH` and `$ORIGIN` work for MQ services and triggered applications. For this reason, for Version 7.1 or later, MQ Services and triggered applications run under the user ID that started the queue manager and not `setuid` or `setgid`.

This information has moved. See UNIX and Linux: MQ services and triggered applications.

UNIX and Linux: `ps -ef | grep amq` interpretation:

From Version 7.1, the interpretation of the list of IBM MQ processes that results from filtering a scan of UNIX or Linux processes has changed. The results can show IBM MQ processes running for multiple installations on a server. Before Version 7.1, the search identified IBM MQ processes running on only a single installation of the product on a UNIX or Linux server.

This information has moved. See UNIX and Linux: `ps -ef | grep amq` interpretation.

UNIX and Linux: `/usr` symbolic links removed:

From Version 7.1, on all UNIX and Linux platforms, the links from the `/usr` file system are no longer made automatically. In order to take advantage of these links, you must set an installation as the primary installation.

This information has moved. See UNIX and Linux: /usr symbolic links removed.

Configurable certificate validation policy on Windows and UNIX: UNIX Windows

From, Version 7.1.0, Fix Pack 2 you configure the product to specify which SSL or TLS certificate validation policy is used to validate digital certificates received from remote partner systems.

This information has moved. See What's changed in IBM WebSphere MQ Version 7.1 Fix Packs.

`amqmsrvn.exe` process removed on Windows: Windows

From Version 7.1, the `amqmsrvn.exe` DCOM process was replaced by a Windows service, `amqsvc.exe`. This change is unlikely to cause any problems. However, you might have to make some changes. You might have configured the user that runs the Windows service `MQSeriesServices` without the user right to `Log on as a service`. Alternatively, the user might not have `List Folder` privilege on all the subdirectories from the root of the drive to the location of the service `amqsvc.exe`.

This information has moved. See Windows:`amqmsrvn.exe` process removed.

IgnoredErrorCodes registry key on Windows: Windows

From Version 7.1, the registry key used to specify error codes that you do not want written to the Windows Application Event Log has changed.

This information has moved. See Windows: IgnoredErrorCodes registry key.

Installation and infrastructure information on Windows: Windows

From Version 7.1, the location of Windows installation and infrastructure information has changed.

This information has moved. See Windows: Installation and infrastructure information.

Local queue performance monitoring on Windows: Windows

On Windows, from Version 7.1, it is no longer possible to monitor local queues using the Windows performance monitor.

This information has moved. See Windows: Local queue performance monitoring.

Logon as a service required on Windows:

From Version 7.1, the user ID that runs the IBM MQ Windows service must have the user privilege to `Logon as a service`. If the user ID does not have the privilege to run the service, the service does not start and it returns an error in the Windows system event log. Typically, you will have run the Prepare IBM MQ wizard, and set up the user ID correctly. Only if you have configured the user ID manually is it possible that you might have a problem in Version 7.1 or later.

This information has moved. See Windows: “Logon as a service” required.

MSCS restriction with multiple installations on Windows: Windows

When you install or upgrade to Version 7.1 or later, the first installation of the product on the server is the only one that can be used with Microsoft Cluster Server (MSCS). No other installations on the server can be used with MSCS. This restriction limits the use of MSCS with multiple installations of the product.

This information has moved. See Windows: MSCS restriction with multiple installations.

Migration of registry information on Windows: Windows

Before Version 7.1, all IBM MQ configuration information, and most queue manager configuration information, was stored in the Windows registry. From Version 7.1 onwards, all queue manager configuration information (for example, `mqs.ini`, `qmstatus.ini`, and `qm.ini`) is stored in files; the same files as in UNIX and Linux platforms.

This information has moved. See Windows: Migration of registry information.

Relocation of the mqclient.ini file on Windows: Windows

From Version 7.1, the `mqclient.ini` file has moved from FilePath to WorkPath. This is similar to the model already used on UNIX and Linux systems.

This information has moved. See Windows: Relocation of the mqclient.ini file.

Task Manager interpretation on Windows: Windows

From Version 7.1, the interpretation of the processes that are listed by the Windows Task Manager has changed. The results can show IBM MQ processes running for multiple installations on a server. Before Version 7.1, the process list identified IBM MQ processes running on only a single installation of the product on a Windows server.

This information has changed. See:
- Windows: Task manager interpretation
- Windows: changes from IBM MQ Version 8.0

IBM MQ Active Directory Services Interface on Windows: Windows

From Version 7.1, the IBM WebSphere MQ Active Directory Services Interface is no longer available.

This information has moved. See Windows: IBM WebSphere MQ Active Directory Services Interface.

GSKit: Changes from GSKit Version 7.0 to Version 8.0: Multi

For Multiplatforms, from IBM WebSphere MQ Version 7.1, GSKit Version 8.0 is integrated with IBM WebSphere MQ and is the only version of GSKit provided with the product. GSKit Version 7.0 is no longer provided.

This information has moved. See GSKit: Changes from GSKit V7.0 to GSKit V8.0.

GSKit: Some FIPS 140-2 compliant channels do not start: Multi

From IBM WebSphere MQ Version 7.1 three CipherSpecs are no longer FIPS 140-2 compliant. If a client or queue manager is configured to require FIPS 140-2 compliance, channels that use the following CipherSpecs do not start after migration.

This information has moved. See GSKit: Some FIPS 140-2 compliant channels do not start.

GSKit: Certificate Common Name (CN) not mandatory: Multi

In GSKit Version 8.0, the **iKeyman** command accepts any element of the distinguished name (DN), or a form of the subject alternative name (SAN). It does not mandate that you provide it with a common name. In GSKit Version 7.0, if you create a self-signed certificate using the **iKeyman** command you had to specify a common name.

This information has moved. See GSKit: Certificate Common Name (CN) not mandatory.

GSKit: Commands renamed: Multi

The command name **gsk7cmd** is replaced with **runmqckm** ; **gsk7ikm** is replaced with **strmqikm**, and **gsk7capicmd** is replaced with **runmqakm**. All the commands start the GSKit Version 8.0 certificate administration tools, and not the GSKit Version 7.0 tools.

This information has moved. See GSKit: Commands renamed.

GSKit: The **iKeyman** *commands to insert a certificate do not check that all required CA certificates are present:* Multi

The **iKeyman** command in GSKit V8.0 does not validate a certificate when it is inserted into a key repository. **iKeyman** in GSKit V7.0, validated a certificate before it inserted the certificate into a certificate store.

This information has moved. See GSKit: The **iKeyman** commands to insert a certificate do not check that all required CA certificates are present.

GSKit: PKCS#11 and IBM MQ JRE addressing mode: Multi

If you use **iKeyman** or **iKeycmd** to administer certificates and keys for PKCS#11 cryptographic hardware note that the addressing mode of the IBM MQ JRE for these tools has changed from IBM WebSphere MQ Version 7.1.

This information has moved. See GSKit: PKCS#11 and IBM MQ JRE addressing mode.

GSKit: Import of a duplicate PKCS#12 certificate: Multi

In GSKit V8.0, the **iKeyman** command does not report an attempt to import a duplicate PKCS#12 certificate as an error. In GSKit V7.0, the **iKeyman** command reported an error. In neither version is a duplicate certificate imported. For GSKIT V8.0, a duplicate certificate is a certificate with the same label and public key.

This information has moved. See GSKit: Import of a duplicate PKCS#12 certificate.

GSKit: Certificate stores created by **iKeyman** *and* **iKeycmd** *no longer contain CA certificates:* Multi

The **iKeyman** and **iKeycmd** utilities in GSKit V8.0 create a certificate store without adding pre-defined CA certificates to the store. To create a working certificate store, you must now add all the certificates that you require and trust. In GSKit V7.0 **iKeyman** and **iKeycmd** created a certificate store that already contained CA certificates. Existing data bases created by GSKit V7.0 are unaffected by this change.

This information has moved. See GSKit: Certificate stores created by **iKeyman** and **iKeycmd** no longer contain CA certificates.

GSKit: Password expiry to key database deprecated: Multi

In GSKit V8.0, the password expiry function in **iKeyman** continues to work the same as in GSKit V7.0, but it might be withdrawn in future versions of GSKit. Use the file system protection provided with the operating system to protect the key database and password stash file.

This information has moved. See GSKit: Password expiry to key database deprecated.

GSKit: Recompile LinuxC++ applications and update run time libraries: Multi

C++ IBM MQ MQI client and server applications on Linux must be recompiled using a supported version of the GNU Compiler Collection (GCC). The C++ run time libraries for this version of GCC must be installed in `/usr/lib` or `/usr/lib64`.

This information has moved. See GSKit: Recompile Linux C++ applications and update run time libraries.

GSKit: Signature algorithm moved out of settings file: Multi

In GSKit V8.0, the default signature algorithm used when creating self-signed certificates or certificate requests or selected in the creation dialogs is passed as a command-line parameter. In GSKit V7.0, the default signature algorithm was specified in the settings file.

This information has moved. See GSKit: Signature algorithm moved out of settings file.

GSKit: Signed certificate validity period not within signer validity: Multi

In GSKit V8.0, the **iKeyman** command does not check whether the validity period of a resulting certificate is within the validity period of the signed certificate. In GSKit V7.0, **iKeyman** checked that the validity period of the resulting certificate was within the validity period of the signed certificate.

This information has moved. See GSKit: Signed certificate validity period not within signer validity.

GSKit: Stricter default file permissions: Multi

The default file permissions set by **runmqckm** and **strmqikm** in IBM WebSphere MQ Version 7.1 or later on UNIX and Linux are stricter than the permissions that are set by **runmqckm**, **strmqikm**, **gsk7cmd**, and **gsk7ikm** in earlier releases of IBM MQ.

This information has moved. See GSKit: Stricter default file permissions.

Migration paths

You can find topics about direct migration to the current release of IBM MQ in this release of the IBM MQ product documentation. Paths between other releases are described in previous releases of the product documentation, which are available on the web.

Multi

Migration paths: IBM MQ for Multiplatforms

Note: Backward migration is not possible.

It is impossible to restore a queue manager from Version 7.0.1 to an earlier version. If a queue manager has not been started, it is possible to uninstall the current version and reinstall a different version of IBM MQ. It does not matter what versions of IBM MQ are installed between when a queue manager was last started and when it is next started.

Table 79. Migration paths: IBM MQ for Multiplatforms

From / To	**Version 7.1, Version 7.5, or Version 8.0**	**Version 9.0**
Prior to Version 7.0.1	Follow the instructions for your current version. For more information about where to find the documentation for older versions of the product, see Documentation for older versions of IBM MQ. **Note:** In the case of migration to Version 8.0, direct migration is not possible. You must follow an indirect migration path, which involves migrating IBM MQ more than once.	Direct migration is not possible. You must follow an indirect migration path, which involves migrating IBM MQ more than once.
Version 7.0.1 or later	Follow the instructions for your current version. See the IBM MQ product documentation in IBM Knowledge Center.	"Migration concepts and methods" on page 645 Windows "Planning to migrate to a later version on Windows" on page 685 Linux UNIX "Planning to migrate to a later version on UNIX and Linux" on page 720 IBM i "Planning to migrate to the latest version on IBM i" on page 749

z/OS

Migration paths: IBM MQ for z/OS

Table 80. Migration paths: IBM MQ for z/OS

From / To	Version 7.1 or Version 8.0	Version 9.0
Prior to Version 7.0.1	Follow the instructions on the web for your current version. See the IBM MQ documentation library web page for further details. **Note:** In the case of Version 8.0, direct migration is not possible. You must follow an indirect migration path, which involves migrating IBM MQ more than once.	Direct migration is not possible. You must follow an indirect migration path, which involves migrating IBM MQ more than once. See "Migrating from earlier unsupported releases of IBM WebSphere MQ for z/OS" on page 789 for further information.
Version 7.0.1 or later	Follow the instructions for your current version. See the IBM MQ product documentation in IBM Knowledge Center.	"Planning to migrate to the latest release on z/OS" on page 772

To revert an IBM MQ for z/OS queue manager to an earlier version, see "Reverting a queue manager to a previous release on z/OS" on page 801.

Migration concepts and methods

An overview of the various concepts and methods for migrating from one release of the product to another.

Objects to consider during migration

It is important to consider four types of object during migration:

Operating environment migration
Upgrading the operating environment, or components in the environment such as installing a new level of JRE; see "IBM MQ operating environment migration" on page 648

Queue manager migration
Migrating a queue manager following an upgrade of the IBM MQ installation to a new command level; see "Queue manager migration" on page 648.

IBM MQ MQI client migration
Migrating a client configuration following installation of a new version or release of the IBM MQ MQI client ; see "IBM MQ MQI client migration" on page 650.

Application migration
Relinking, recompiling, or recoding an IBM MQ server or client application; see "Application migration and interoperation" on page 652. Application migration also includes migrating any API or channel exits

Impact of migration on other queue managers or clients

In addition, you must consider the impact of migrating one queue manager, or IBM MQ MQI client, on other queue managers or clients:

Compatibility, coexistence, and interoperability
See "Coexistence, compatibility, and interoperability" on page 663 for information about the compatibility of IBM MQ applications connected to queue managers and IBM MQ MQI client clients on different command levels. The section also explains the concept of queue manager coexistence, and the interoperability of IBM MQ JMS applications with WebSphere Application Server.

Queue manager clusters

Can a queue manager cluster contain queue managers at different command levels? See "Migrating a queue manager cluster" on page 808 to answer this question, and how to migrate a cluster of queue managers.

z/OS **Queue-sharing groups**

Queue-sharing groups involve multiple queue managers running on z/OS. How do you migrate queue managers that are part of a queue-sharing group to a new command level; see "Queue-sharing group migration" on page 805.

High-availability clusters

How do you migrate queue managers that are part of a high-availability cluster to a new command level, and maintain continuous and reliable service? See "Migrating a queue manager in a high-availability configuration" on page 814, which covers both migration of multi-instance queue managers, and the migration of queue managers operating in high-availability clusters.

IBM MQ application migration model

Figure 61 on page 647 shows two runtime operating system environments. One environment is called `Server`, and contains an IBM MQ server and server application. The other is called `Client`, and contains an IBM MQ MQI client application. The server environment has one or more queue managers represented by **`QM`** using the installation of IBM MQ installed on the server.

The queue manager labeled `QM-n?` coexists on the same server as `QM`, but runs at a different release level. Multiple releases of IBM MQ installed in the same operating environment are called coexistent [2] . The IBM MQ installations for different release levels are not shown. The question-mark in the queue manager name indicates this capability might not be present in your environment.

z/OS Only z/OS supports multiple queue managers coexisting at different release levels in the same operating environment.

Queue manager coexistence is important for migration in two respects:

1. It can be used to reduce the risk involved in migrating to a new command level, and reduce the downtime during the migration process.
2. You must consider any configuration implications of running some applications or clusters on the same server with queue managers at different command levels.

For more information, see "Queue manager coexistence" on page 664.

The queue manager, `QM*`, represents queue managers of various levels installed on other servers.

The following diagram shows a client and server, each of which contains a number of software components, such as databases, application servers, and the language or subsystem run time environment. The environment contains an IBM MQ application, the IBM MQ MQI client or server library, and IBM MQ channel and API exit programs. These components are connected to a queue manager component, either locally in the server, or remotely to the same server queue manager from the client. The application is linked to the IBM MQ library by the MQI. The libraries are shown linked to the queue manager either by an SPI, which describes the connection between the process running the MQI and the queue manager processes, or by an IBM MQ MQI client connection. The diagram also shows the queue manager linked to another queue manager at a different level on another server, and also a queue manager, QM-n, on the same server. The queue manager called QM-n is running at a lower level. It represents a number of queue managers of different versions, coexisting on the same server.

2. It is not necessary, but it is usual, for coexistent installations to be at different release levels.

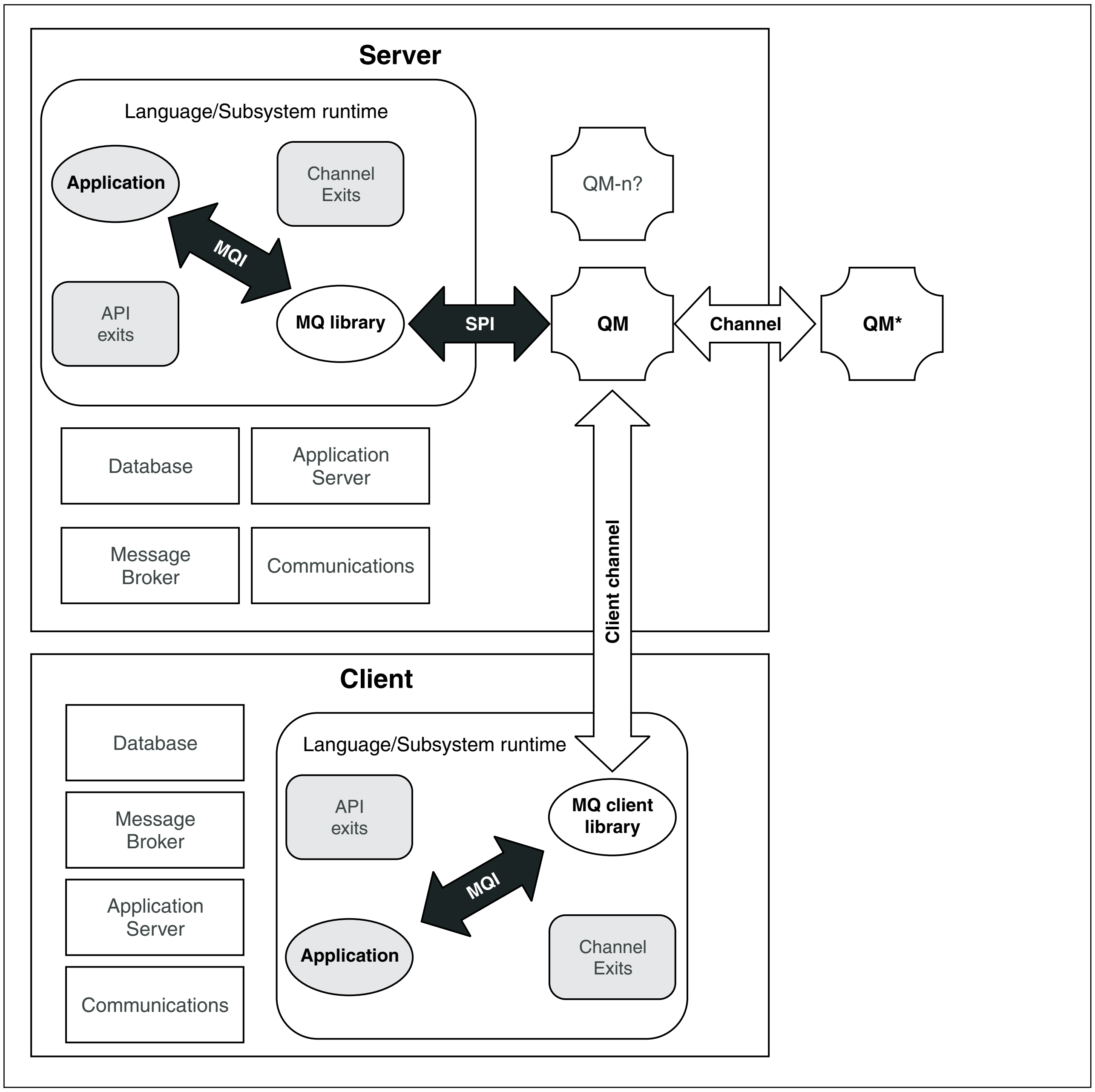


Figure 61. IBM MQ application migration model

IBM MQ operating environment migration

You might need to perform some migration tasks for IBM MQ as a result of upgrading the operating environment.

To find out what operating environment upgrades you must make before upgrading IBM MQ, compare the requirements for different releases. For more information about system requirements, see System Requirements for IBM MQ.

Note that the System Requirements page for IBM MQ Version 9.0 uses the Software Product Compatibility Reports (SPCR) tool.

By selecting the appropriate link on the web page, the SPCR tool enables you to go directly to the following information for the specific operating system, or systems, that your enterprise uses.

- Supported operating systems
- Prerequisites
- System requirements
- Optional supported software

For details about operating environment changes in the latest release that directly affect the migration to a new version of IBM MQ, see the following information:

- V 9.0.0 What's new and changed in IBM MQ Version 9.0
- CD What's new and changed in Version 9.0.x Continuous Delivery
- LTS What's changed in Version 9.0.0.x Long Term Support
- LTS CD Readme for IBM MQ V9.0 and its maintenance

For information about what changed in an earlier release of the product, see the *What's changed* section in the product documentation for that release:

- What's changed in IBM MQ Version 8.0
- What's changed in IBM WebSphere MQ Version 7.5
- What's changed in IBM WebSphere MQ Version 7.1
- What's changed in IBM WebSphere MQ Version 7.0.1

Some changes might affect IBM MQ migration indirectly. For example, the runtime linkage conventions for applications, or the way memory is allocated, might change.

Queue manager migration

After upgrading an installation, queue manager migration might be required. Migration takes place when you start a queue manager. You can remove an upgrade before you have started a queue manager. However, if you remove the upgrade after a queue manager has been started, the queue manager will not work.

Migrating a queue manager to a later release

z/OS On IBM MQ for z/OS, queue manager migration is required after upgrading to a different version, release, or maintenance level of the product. The upgrade changes the command level. The current command, or VRM, level is shown in the z/OS console log.

Multi On IBM MQ for Multiplatforms, queue manager migration is always required for changes in the first two digits of the VRMF code. Changes in the maintenance and fix level, M and F in the VRMF code, never cause automatic queue manager migration. No migration was required for the upgrade from Version 7.0 to Version 7.0.1. The change from Version 7.0 to Version 7.0.1 did change the command level from 700 to 701. From Version 7.1 onwards, a change in the command level always requires queue manager migration, but if the change is shipped in a maintenance or fix pack, you have the choice of whether to increase the command level, and cause queue manager migration.

Command level always increases with a change in version or release. If you decide to use new function introduced in a maintenance level upgrade, you must change the command level. The converse is not the case. You do not have to change the command level when the fix level changes. You can decide to install the fix pack, but not use the new function. Whether or not you use the new function, the installation of the fix pack increases the maximum command level supported by the installation. Run the **dspmqver** command to display the current maximum supported command level.

Queue manager migration is the process of converting persistent queue manager data from one version to another. Persistent queue manager data includes log files and data in the queue manager directory. The data records changes to objects such as messages, subscriptions, publications, queue managers, channels, queues, and topics.

Queue manager migration is required and largely automatic.

z/OS On z/OS, you must migrate queue managers manually between compatibility mode and new function mode by setting the **OPMODE** parameter.

You can reduce the downtime and risk caused by queue manager migration, by verifying the new version first, using a different queue manager. Unless the platform supports queue manager coexistence, you need to perform the verification on a different server, or in a virtualized environment on the same server. If the platform you are upgrading supports queue manager coexistence, you can install the new version of IBM MQ on the same server, verify it, and minimize downtime to the time required to stop, backup, and restart the queue manager.

Note: If you are migrating a queue manager through multiple release levels, one level at a time, you must start the queue manager after each upgrade to migrate it. You must also start all the channels, to ensure they are migrated.

Restoring a queue manager to an earlier release

Multi For IBM MQ for Multiplatforms, you cannot restore a queue manager to an earlier release level after you have migrated it to a new release. You must back up your system before starting backwards migration. You can either back up queue manager data, or use a backup queue manager; see Backing up and restoring IBM MQ. Before backing up, you must stop the queue manager.

z/OS For IBM MQ for z/OS, the following considerations apply to migration:

- Before Version 7.0.1, it is possible to revert to an earlier level, as long as you have applied the correct PTFs. However, from Version 7.0.1 onwards, it is impossible to revert to an earlier release after switching a queue manager to new function mode by running with **OPMODE** *NEWFUNC*. Provided that you have not switched the queue manager to new function mode, you can backwards migrate, as described in Migration PTFs.
- LTS From Version 9.0, you can backwards migrate queue managers only if you are using the Long Term Support (LTS) release model, and provided that you have not set the **OPMODE** to NEWFUNC. For more information, see IBM MQ release types.
- On z/OS, you must migrate queue managers manually between compatibility mode and new function mode by setting the **OPMODE** parameter. If you have never switched a queue manager to new function mode, you can still run it against the earliest release it is compatible with. You must have applied compatibility PTFs to the earlier release before starting a queue manager at the new command level. The compatibility level is shown in the log.

Related concepts:

Multi "The version naming scheme for IBM MQ for Multiplatforms" on page 567
From IBM MQ Version 9.0, releases have a three digit Version, Release, and Modification (VRM) code or a four-digit Version, Release, Maintenance, and Fix (VRMF) level code.

z/OS "The version naming scheme for IBM MQ for z/OS" on page 569
On IBM MQ for z/OS, releases have a three digit Version, Release, and Modification (VRM) code. To run a queue manager at a different VRM level, you must migrate the queue manager, its applications, and the environment in which it runs. Depending on the migration path, the migration might require more or less effort.

z/OS "Upgrade and migration of IBM MQ on z/OS" on page 785
You can install new releases of IBM MQ to upgrade IBM MQ to a new release, or version level. Multiple installations at the same or different levels can coexist on the same z/OS instance. Running a queue manager at a higher level requires migration.

Related tasks:

UNIX Linux "Migrating a queue manager on UNIX and Linux" on page 721
The procedures for migrating a queue manager to a later version of the product, and for restoring a queue manager to an earlier version of the product are detailed in this section.

Windows "Migrating a queue manager on Windows" on page 691
The procedures for migrating a queue manager to a later version of the product, and for restoring a queue manager to an earlier version of the product are detailed in this section.

IBM i "Migrating a queue manager to a later version on IBM i" on page 751
Follow these instructions to migrate a queue manager from an earlier release to a later release.

Related reference:

z/OS "OPMODE on z/OS" on page 781
The availability of new functions and backward migration for IBM MQ for z/OS is controlled by the **OPMODE** parameter in the **CSQ6SYSP** macro. IBM MQ Version 8.0 new functions that are restricted by **OPMODE** are not available at Version 9.0 unless enabled with **OPMODE**. There are no new functions in Version 9.0 that are restricted by **OPMODE**.

Related information:

Restoration of a queue manager to an earlier version on all platforms:

You can remove an upgrade before you have started a queue manager. However, if you remove the upgrade after a queue manager has been started, the queue manager will not work.

This information has moved. See "Queue manager migration" on page 648.

IBM MQ MQI client migration

IBM MQ MQI client migration is the process of converting IBM MQ MQI client configurations, and client and server channels from one version to another. Client migration can take place after upgrading the IBM MQ MQI client, and is reversible.

Client migration on the client workstation is optional and manual. Client migration on the server is required and automatic. See "Changes that affect migration" on page 626 for links to further information about any client changes. You must upgrade an IBM MQ MQI client before migrating a client workstation to make use of new configuration options. You can make configuration changes to client and server connection channels on the server, but they have no effect on a client workstation, until the client is upgraded.

An example of client migration performed at the client workstation is to manually migrate configuration settings to the `mqclient.ini` configuration file.

An example of combined client and server migration is the deployment of a new client connection definition table (CCDT). To use a new version of the CCDT, generate the table on a queue manager that is at the new code level. Deploy the table to clients that are going to use it. To deploy the table to a client, you first must update the client to at least the same level as the queue manager that created the table.

An IBM MQ MQI client can interoperate with earlier and later versions of IBM MQ. Upgrading the IBM MQ MQI client makes new function available to client applications, and is important to maintain the service level. Migrating an IBM MQ MQI client gives it access to new configuration options.

The IBM MQ MQI client libraries, such as `mqic.dll`, are dynamic, and the application linkages to the libraries do not normally change. You do not relink a client application to pick up new IBM MQ client libraries. The client picks up the new library next time the library is loaded by the client application. Do not move libraries from their installed directory. Linking to libraries in anything other than their installed directory is an unsupported configuration.

Related concepts:

"Application compatibility and interoperability with earlier versions of IBM MQ" on page 676
Connecting an application that is built against libraries shipped with a later version of IBM MQ to an earlier version IBM MQ is not supported. Avoid building applications against a later version, and redeploying them to a queue manager running at an earlier version, although some applications do work in practice.

"Application compatibility and interoperability with later versions of IBM MQ" on page 678
IBM MQ applications run against later versions of a queue manager without recoding, recompiling, or relinking. You can connect an application that is built against libraries shipped with an earlier version of IBM MQ to a queue manager running at a later version of IBM MQ.

Related tasks:

IBM i "Migrating an IBM MQ MQI client to the latest version on IBM i" on page 765
Before migrating an IBM MQ MQI client, create a migration plan. Stop all IBM MQ activity on the client workstation. Upgrade the IBM MQ MQI client installation. Make any essential configuration and application changes.

UNIX Linux "Migrating an IBM MQ MQI client on UNIX and Linux" on page 736
Before migrating an IBM MQ MQI client, create a migration plan. Stop all IBM MQ activity on the client workstation. Upgrade the IBM MQ MQI client installation. Make any essential configuration and application changes.

Windows "Migrating an IBM MQ MQI client on Windows" on page 707
Before migrating an IBM MQ MQI client, create a migration plan. Stop all IBM MQ activity on the client workstation. Upgrade the IBM MQ MQI client installation. Make any essential configuration and application changes.

Related reference:

"MQI client: Client Channel Definition Table (CCDT)" on page 680
You can connect an IBM MQ MQI client application to any level of queue manager. If a client uses CCDT to connect to a queue manager, the CCDT can be at a version greater than, less than, or equal to that of the client.

"MQI client: Default behavior of client-connection and server-connection channels" on page 681
In Version 7.0 the default settings for client and server connection channels changed to use shared conversations. This change affects the behavior of heartbeats and channels exits, and can have an impact on performance.

"MQI client: MQPUT1 sync point behavior change" on page 683
An MQPUT1 call by an IBM MQ MQI client application that failed in IBM WebSphere MQ Version 6.0 can now sometimes succeed. The failure is returned to the application later, if it calls MQCMIT. For the

change in behavior to occur, the MQPUT1 must be in sync point.

Application migration and interoperation

IBM MQ supports running applications compiled and linked against previous versions of IBM MQ, with later levels of IBM MQ. IBM MQ applications might require migration between Version 7.1 and the latest version.

To migrate an application to run with a new level of IBM MQ, disconnect an application from the queue manager. Reconnect it when the queue manager is running again. However, it takes only one small difference in the interface between IBM MQ and the application to break an application, or make it behave wrongly. Sometimes a problem does not show up for a long time. For this reason, you must always test your applications against a new version of IBM MQ. The suggested extent of testing varies depending on the extent of the changes in IBM MQ ; see “Characteristics of different types of upgrade on z/OS” on page 786 or “Characteristics of different types of upgrade” on page 619.

Application migration refers to four kinds of changes.

1. Application changes that are consequent to upgrading the operating environment along with the queue manager. Rarely, linkage conventions change. The most likely reason for a linkage change is switching from 32 bit to a 64 bit environment. If you are using SSL or TLS you might have to relink with a new secure library.
2. Changes that you must make to the application in order to run an application against a new level of queue manager. Changes of this sort are uncommon. However, you must check “Changes that affect migration” on page 626 to see if any changes might affect your applications.
3. Changes that are not required, but that you might want to make in future, perhaps because you have a business reason to modify an application.
4. Changes to applications that are supplied by IBM, or other vendors, that require you to run migration utilities. The utilities convert the applications to running on the new version of IBM MQ.

Do not load IBM MQ libraries from an earlier level. IBM MQ does not support connecting server applications loading libraries from the earlier level to connect to a later level of queue manager. On UNIX, Linux, and Windows platforms, the application load path must be set up to the location of the IBM MQ server libraries. You do not have to recompile and relink an application. Applications compiled and linked against an earlier version of IBM MQ can load libraries from a later version.

Multi On Multiplatforms, from Version 7.1 onwards, IBM MQ loads the library from the installation the application is connecting to. An application must initially load a library of at least the same level as the application linked to. IBM MQ then loads the correct version of the library from the installation that the queue manager is associated with. If you have two installations of the same version, but at different fix levels, IBM MQ chooses which library to load. The choice is based on the queue manager the application is connected to. If an application is connected to multiple queue managers, it is possible that multiple libraries are loaded.

To help you write applications that can exchange messages with earlier versions of the product, IBM MQ provides data type versioning. Data type versioning assists you in exchanging messages that are compatible with target queue managers. A good programming practice is to set the version number of a data structure explicitly. Do not assume that the default version is the one you require. By setting the version explicitly, you are forced to look up what version to use. The description of the data type version tells you what level of queue manager supports that version.

It is poor practice to set the data type version to the current version. If you recompile your program against a new version of IBM MQ, the data type version might change with unexpected consequences.

Client applications are more likely to connect to different queue managers than applications written for a specific server. Plan carefully when writing an application that is to connect to different versions of a queue manager, and to queue managers on different platforms. The default values of some IBM MQ constants, such as `MQPMO_SYNCPOINT`, `MQPMO_NO_SYNCPOINT` differ between platforms. Some functions are not available on all platforms.

You must be aware of, and code to, the capabilities of all the queue managers the application interacts with. It requires planning and design to write an application that works with different versions of a queue manager. There is no API provided with IBM MQ to restrict an application to a function subset common to the set of queue managers it interacts with. To improve interoperability, some developers choose to provide an MQI wrapper layer, or use MQI API exits, to control the functions programs use.

Connection authentication

For a new IBM MQ Version 8.0, or later, installation, the **`CONNAUTH CHCKLOCL`** attribute will be set to `OPTIONAL`. This means that user IDs and passwords are not required, but if they are provided they must be a valid pair, or they will be rejected.

When you are migrating between IBM WebSphere MQ Version 7.1 and the latest version, the **`CONNAUTH CHCKLOCL`** attribute on each queue manager is set to `NONE`, ensuring version to version continuity, but switching connection authentication off.

For more information see Connection authentication: Configuration.

Related concepts:

"Application compatibility and interoperability with earlier versions of IBM MQ" on page 676
Connecting an application that is built against libraries shipped with a later version of IBM MQ to an earlier version IBM MQ is not supported. Avoid building applications against a later version, and redeploying them to a queue manager running at an earlier version, although some applications do work in practice.

"Application compatibility and interoperability with later versions of IBM MQ" on page 678
IBM MQ applications run against later versions of a queue manager without recoding, recompiling, or relinking. You can connect an application that is built against libraries shipped with an earlier version of IBM MQ to a queue manager running at a later version of IBM MQ.

Related tasks:

UNIX Linux "Migrating IBM MQ library loading to a later version on UNIX and Linux" on page 739
On UNIX and Linux, no change in the way IBM MQ libraries are loaded is normally required if you upgrade from an earlier version of the product to a later version by replacing an earlier version of the product with the later version, based on the single stage scenario. However, if you choose to take advantage of multi-installation in the later version of the product, based on the side-by-side or multi-stage migration scenarios, you might have to configure the runtime environment differently, for the operating system to load the later version of the IBM MQ library.

Windows "Migrating IBM MQ library loading to a later version on Windows" on page 709
On Windows, no change in the way IBM MQ libraries are loaded is normally required if you upgrade from an earlier version of the product to the later version by replacing an earlier version of the product with the later version, based on the single stage scenario. However, if you choose to take advantage of multi-installation in the later version of the product, based on the side-by-side or multi-stage migration scenarios, you might have to configure the runtime environment differently, for the operating system to load the later version of the IBM MQ library.

Migration methods on IBM MQ for Multiplatforms

Multi

There are three main methods of migrating from one release to another: Single-stage migration (called a slip installation on IBM i), side-by-side migration, and multi-stage migration. Multi-stage migration is not an option for IBM i.

Single-stage migration

Single-stage migration is the term that is used to describe replacing the only installation of IBM MQ on a server, with a later release. Before IBM WebSphere MQ Version 7.1, single-stage was the only migration scenario.

The advantage of single-stage migration is that it changes the configuration of a queue manager on the earlier version as little as possible. Existing applications switch from loading the libraries from the earlier version, to loading the libraries of the later version, automatically. Queue managers are automatically associated with the installation on the later version. Administrative scripts and procedures are affected as little as possible by setting the installation to be the primary installation. If you set the installation of the later version to be the primary installation, commands such as **strmqm** work without providing an explicit path to the command.

Of the three approaches, single-stage migration preserves the greatest number of existing scripts and procedures for running IBM MQ. However, the other migration approaches support a gentler transition to the new version, which can reduce the overall impact on users.

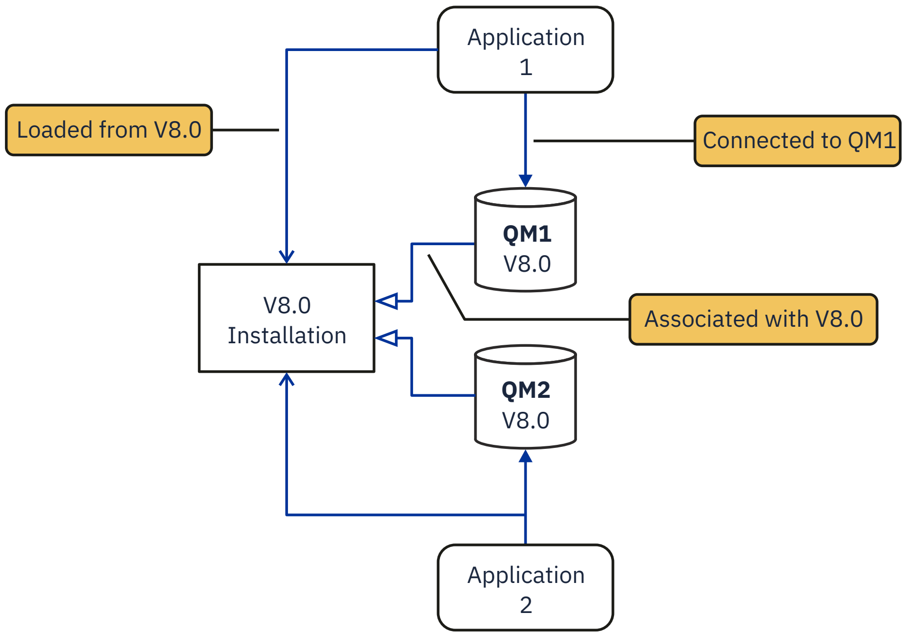


Figure 62. Single_stage migration: earlier version installed with connected queue managers and associated applications

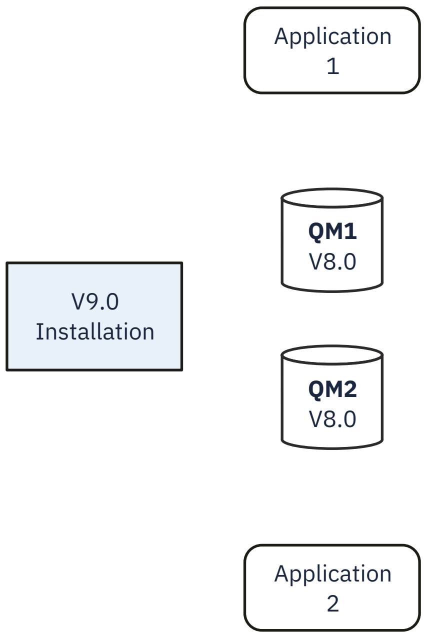


Figure 63. Single_stage migration: later version installed but queue managers not yet connected and applications not yet associated

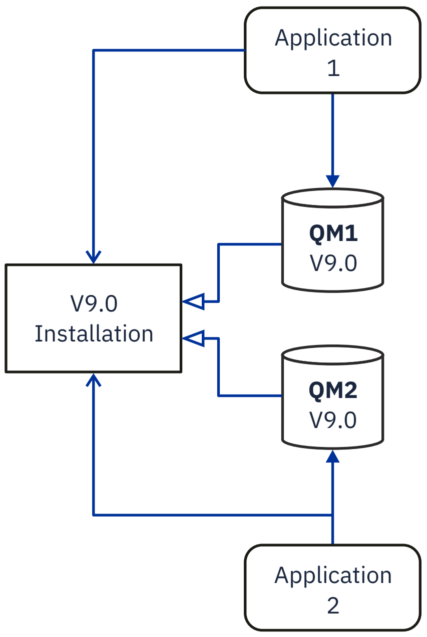


Figure 64. Single_stage migration: migrated queue managers connected to and applications associated with later version

For more information about single-stage migration, see:
- UNIX Linux "Migrating on UNIX and Linux: single-stage" on page 724
- Windows "Migrating on Windows: single stage" on page 694

- IBM i "Installation methods on IBM i" on page 752 (on IBM i, a single-stage migration is called a slip installation)

Side-by-side migration

On UNIX, Linux and Windows, side-by-side migration is the term that is used to describe installing a later version of IBM MQ alongside an older version on the same server. The side-by-side migration scenario sits half-way between the single-stage and multi-stage migration scenarios and is based on the following premise:

- Install additional IBM MQ code alongside existing installation while queue managers are still running.
- Move queue managers one at a time to the new installation.
- Migrate and test applications one at a time.

During the installation and verification of the later version of IBM MQ, queue managers continue running, and remain associated with the older version of IBM MQ.

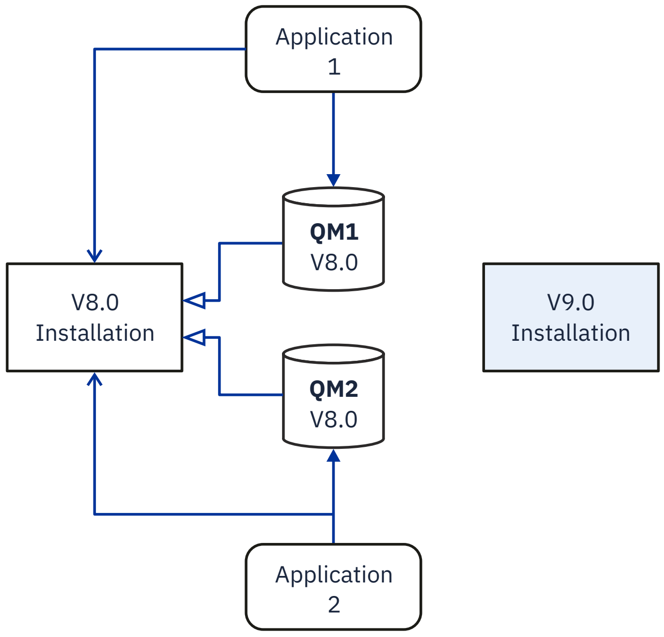


Figure 65. Side-by-side migration: later version installed but queue managers still connected to and applications still associated with earlier version

When you decide to migrate queue managers to the later version of IBM MQ, you stop all queue managers, migrate them all to the later version, and uninstall the earlier version of IBM MQ.

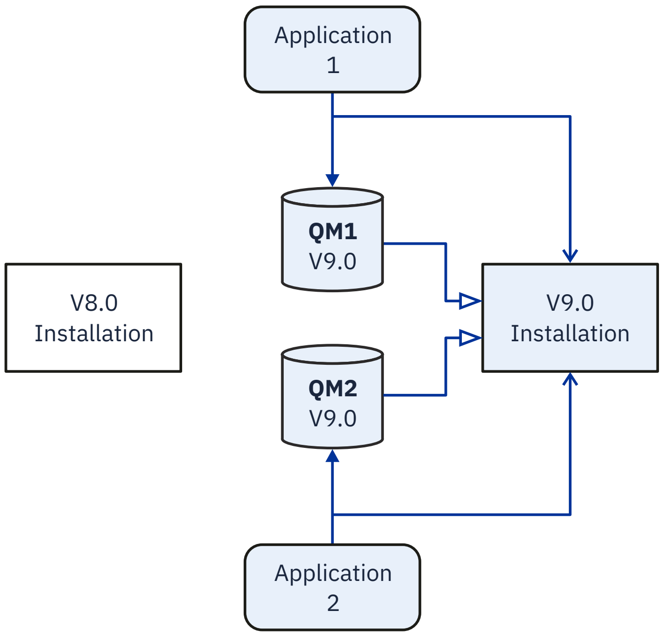


Figure 66. Side-by-side migration: migrated queue managers connected to and applications associated with later version

The advantage that the side-by-side migration has over the single-stage migration is that you can install and verify the later IBM MQ installation on the server before you switch over to it.

Although side-by-side migration is less flexible than multi-stage migration, it does have some advantages over the multi-stage approach. With the side-by-side approach, you can assign a later version of IBM MQ to be the primary installation, whereas, with the multistage approach, you cannot do so, if IBM WebSphere MQ Version 7.0.1 is installed, until you uninstall IBM WebSphere MQ Version 7.0.1. With a later version of IBM MQ set as the primary installation, many applications restart without having to reconfigure their environment, and IBM MQ commands work without providing a local search path.

For more information about side-by-side migration, see:
- UNIX Linux "Migrating on UNIX and Linux: side-by side" on page 727
- Windows "Migrating on Windows: side-by-side" on page 699

Note: IBM i Side-by-side migration has a different meaning on IBM i. A side-by-side installation upgrades IBM MQ on a different computer. For more information, see "Installation methods on IBM i" on page 752. Multiple installations are not applicable to IBM i.

ULW
Multi-stage migration

Multi-stage migration is the term that is used to describe running a later version of IBM MQ alongside an older version on the same server. Multi-stage migration is the most flexible approach.

After you install the later version alongside the earlier version, you can create new queue managers to verify the installation of the later version, and develop new applications. At the same time, you can migrate queue managers and their associated applications from the earlier version to the later version. By migrating queue managers and applications one by one, you can reduce the peak workload on your staff who are managing the migration.

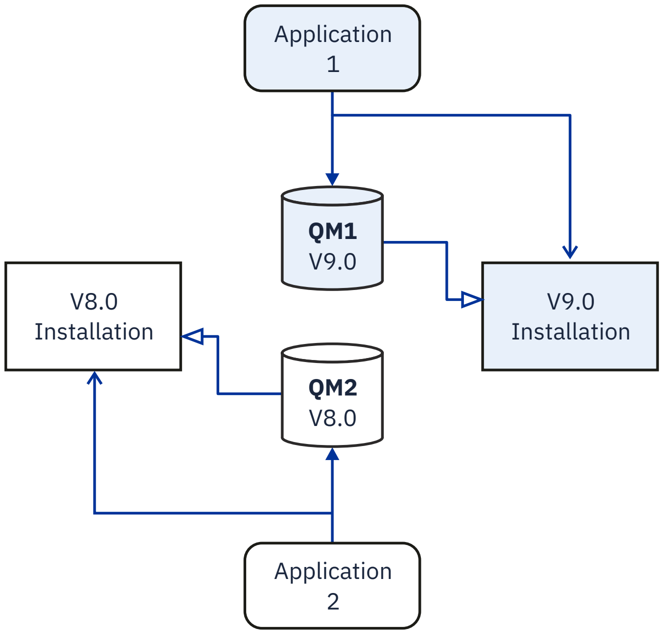


Figure 67. Multi-stage migration: one queue manager and application migrated to later version, and another queue manager and application still at earlier version

For more information about multi-stage migration, see:

- UNIX Linux "Migrating on UNIX and Linux: multi-stage" on page 730
- Windows "Migrating on Windows: multi-stage" on page 701

Related concepts:

z/OS "Upgrade and migration of IBM MQ on z/OS" on page 785
You can install new releases of IBM MQ to upgrade IBM MQ to a new release, or version level. Multiple installations at the same or different levels can coexist on the same z/OS instance. Running a queue manager at a higher level requires migration.

Primary installation on UNIX, Linux and Windows

ULW

On UNIX, Linux, and Windows, which support multiple installations of IBM MQ, you can optionally define one installation as the primary installation. The primary installation is the one to which IBM MQ system-wide locations refer.

Overview

From Version 7.1, you can install multiple versions of the product on UNIX, Linux, and Windows, and configure one of these installations as the primary installation. The primary installation is:

- The installation to which system-wide locations refer
- Optional, but convenient

Before Version 7.1, only one instance of the product could be installed at any time, and the only installation was the primary installation.

You cannot change the primary installation while Version 7.0.1 is installed. However, if you uninstall Version 7.0.1, and then install Version 7.1 or later, the new installation is not by default the primary installation. If all the installations on the system are at Version 7.1, or later, you can choose whether to have a primary installation.

UNIX Linux

UNIX and Linux

The primary installation:

- Has symbolic links in `/usr/lib` and `/usr/bin`

  If you have not set a primary installation there are no symbolic links.
- Must be configured manually using the following command:

```
$ MQ_INSTALLATION_PATH/bin/setmqinst -i -p MQ_INSTALLATION_PATH
```

To locate your various installations, you can:

- Use the platform installation tools to query what is installed and where on the system
- Use the dspmqver command to display IBM MQ version and build information.
- Use the dspmqinst command to display installation entries from `mqinst.ini`.
- Use the following command to list the installations:

```
cat /etc/opt/mqm/mqinst.ini
```

Windows

Windows

The primary installation is:

- By default the first installation.
- Pointed to by global environment variables.
- Used by some operating system features that require central registration of interface libraries.

  For example, .NET monitor (transactional mode) and COM/ActiveX interface classes.

To locate your various installations, you can use the:

- Use the platform installation tools to query what is installed and where on the system
- Use the dspmqver command to display IBM MQ version and build information.
- Use the dspmqinst command to display installation entries from `mqinst.ini`.
- Use the following command to query the registry:

```
reg.exe query "HKLM\Software\[Wow6432Node\]IBM\WebSphere MQ\Installation" /s
```

Windows

Migration of Windows registry information

IBM WebSphere MQ Version 7.1 onwards uses `mqs.ini` and `qm.ini`.

However, if IBM WebSphere MQ Version 7.0.1 is still installed, IBM WebSphere MQ Version 7.1 or later still uses the registry.

To uninstall IBM WebSphere MQ Version 7.0.1 with IBM WebSphere MQ Version 7.1 or later already installed, you must complete the following steps:

- End all the queue managers and listeners.
- Uninstall IBM WebSphere MQ Version 7.0.1, which migrates the registry information into `mqs.ini` and `qm.ini`.
- IBM WebSphere MQ Version 7.1 now starts using `mqs.ini` instead.

Note: If for some reason the uninstallation process is interrupted, and the automatic migration does not run, a tool is provided to migrate the information out of the registry manually.

Related information:

Primary installation

Multiple IBM MQ installations

ULW z/OS

From IBM WebSphere MQ Version 7.1, multiple IBM MQ installations are supported on UNIX, Linux, and Windows. This gives you the option to install and select between one or more IBM MQ installations.

Overview

You can select between:

- Simplicity of maintaining a single IBM MQ installation.
- Flexibility, by allowing up to a maximum of 128 IBM MQ installations on a system. The first of these 128 installs is reserved for an (optional) IBM WebSphere MQ Version 7.0.1 installation of level Version 7.0.1, Fix Pack 6 or above. The other installations must be IBM WebSphere MQ Version 7.1 or later.

You can install multiple copies of the same code level; this is especially convenient for maintenance purposes.

LTS For example, if you want to upgrade IBM MQ Version 9.0.0.0 to IBM MQ Version 9.0.0, Fix Pack 1, you can install a second copy of IBM MQ Version 9.0.0.0, apply the maintenance to bring it to IBM MQ Version 9.0.0, Fix Pack 1, and then move the queue managers across to the new installation.

You still have the original installation, so it is a simple matter to move the queue managers back if you encounter any problems.

Notes:

1. Linux Solaris On Linux and Solaris only, you must ensure that each package installed has a unique name.

   You need to use a tool to create a unique set of packages:

   - `$ crtmqpkg PACKAGE_SUFFIX`
   - This takes the IBM MQ installation packages, and repackages them with a new name of your choice. You then install as usual.
2. All installations share a data directory; this is where `mqs.ini` is located for example.
3. All installations share the same namespace for queue managers. This means that you cannot create several queue managers of the same name in different installations.
4. IBM MQ installations are fully relocatable; each installation has a separate installation path. You can choose where you would like to install IBM MQ.
5. IBM MQ resources have installation-scope resource isolation, so operations on one installation do not affect the others.

   This means that the resources created by one installation are isolated from those created by other installations. It enables actions, such as removing an installation of IBM MQ, while queue managers are running under another installation.
6. Queue managers are "associated" with an installation You can move them, but you cannot migrate data back to earlier releases.

Working with multiple installations

To work with a queue manager, you need to use the commands from its installation. If you select the wrong installation, you see:

```
AMQ5691: Queue manager 'MYQM' is associated with a different installation (Inst1)
```

To work with a queue manager, you have to use the control commands from its associated installation. You have a choice of:

- Using the full path to the control commands, for example:

  ```
  $ MQ_INSTALLATION_PATH\bin\strmqm MYQM
  ```

  or
- Setting the environment variables for an installation with one of:

  ```
  $ MQ_INSTALLATION_PATH/bin/setmqenv 's
  $ setmqenv -m MYQM
  $ setmqenv -n InstallationName
  $ setmqenv -p MQ_INSTALLATION_PATH
  ```

You might consider using a shell script or batch file to set up the environment for each IBM MQ installation. You can use the **setmqenv** or **crtmqenv** commands to help with this.

- setmqenv sets the values of the environment variables, such as PATH, CLASSPATH and LD_LIBRARY_PATH, for use with an IBM MQ installation.
- crtmqenv creates a list of the environment variables and their values for use with a particular IBM MQ installation. You can then use this list to incorporate into a shell script or batch file.

Commands

To run a command, the operating system must find the command in an IBM MQ installation. In general, you must run a command from the installation that is associated with the correct queue manager. IBM MQ does not switch commands to the correct installation. However, there are some exceptions, such as the **setmqinst** command, where you can run the command from any installation that has the latest version of the product installed.

Commands that work across installations

- dspmq (display queue managers)
- dspmqinst (display IBM MQ installation)
- dspmqver (display version information)
- setmqinst (set IBM MQ installation)

Other control commands for multiple installations

- crtmqenv (create IBM MQ environment)
- dspmqinst (display IBM MQ installation)
- setmqenv (set IBM MQ environment)
- setmqinst (set IBM MQ installation)
- setmqm (set queue manager)

If an earlier version of the product is installed, the command that is run is the command for that version, unless the search path is overridden by a local setting. You can override the search path by running **setmqenv**. If Version 7.0.1 is not installed, you must set the correct path to run a command. If you have set a primary installation, the command that is run is the copy in the primary installation, unless you override the selection with a local search path.

z/OS

Multiple releases on z/OS

Multiple releases can exist on z/OS. You use STEPLIBs to control which level of IBM MQ is used. For more information, see "Coexistence" on page 664.

Related information:
Multiple installations

Multiple installations and application programs: ULW

When a local application connects to a queue manager, the application needs to load the libraries from the installation associated with the queue manager. Multiple installations introduce some complexity.

Using the `setmqm` command

When you use setmqm 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 owned by different installations, multiple sets of libraries need to be loaded.

Note: If you link your applications to IBM WebSphere MQ Version 7.1 or later libraries, the applications automatically load the appropriate libraries when the application connects to a queue manager.

Loading IBM MQ libraries in a multi-version environment

How libraries are located depends upon your environment.

If IBM WebSphere MQ Version 7.1 or later is installed in the default location, existing applications continue to work as before. Otherwise, you might need to rebuild the application or change your configuration.

The order in which libraries are searched, depends upon the platform you are using:
- Windows
  - The application's directory
  - The current directory
  - The global and your PATH variables
- Other platforms
  - LD_LIBRARY_PATH (or LIBPATH/SHLIB_PATH)
  - An embedded search path (RPath)
  - The default library path

Table 81. Options for loading libraries

Platform	Option	Benefits	Drawbacks
UNIX	Set/change the embedded runtime search path (RPath)	The path is explicit in the way the application is built	You need to recompile and link If you move IBM MQ, you must change RPath

Table 81. Options for loading libraries (continued)

Platform	Option	Benefits	Drawbacks
UNIX	Set LD_LIBRARY_PATH or equivalent using setmqenv	Overrides RPath No changes to existing applications Easy to change if you move IBM MQ	Depends on environment variables Possible impacts on other applications
Windows	Set PATH using setmqenv	No changes to existing applications Easy to change if you move IBM MQ	Depends on environment variables Possible impacts on other applications
All	Set the primary installation to IBM WebSphere MQ Version 7.1 or later	No changes to existing applications Easy to change the primary installation Similar behavior to previous versions of IBM MQ	While IBM WebSphere MQ Version 7.0.1 is installed, you cannot make IBM WebSphere MQ Version 7.1 the primary installation UNIX: Relies on `/usr/lib` in the default search path

Related information:

Multiple installations

Coexistence, compatibility, and interoperability

The definitions of the IBM MQ terms coexistence, compatibility, and interoperability.

Coexistence
: Is being able to install and run two or more versions of the same program on the same server. For IBM MQ, it normally means installing and running multiple versions of IBM MQ on a server.

Compatibility
: Is the ability to run applications from one level of queue manager with an earlier, or previous level, of the queue manager.

  If you are using a message channel agent (MCA) channel, any version and release of an IBM MQ queue manager can connect, using an MCA channel, to any version and release of another IBM MQ queue manager.

  The MCA channel is automatically configured to the latest version of protocol that is supported by both ends of the channel.

  Compatibility is also the ability to run client applications with different versions of the IBM MQ MQI client, and different levels of the queue manager.

Interoperability
: Is mainly the ability to exchange messages between different versions of IBM MQ. It can also mean the interoperability between others things, such as publish/subscribe brokers, or between components such as the IBM MQ classes for JMS and WebSphere Application Server.

Maintaining the compatibility, coexistence, and interoperability of IBM MQ is important in order to preserve the investment you make in applications and administrative procedures.

Three areas to which this objective does not apply to as rigidly, are:

- GUI interfaces, such as IBM MQ Explorer.
- Information for service, such as FFST files and traces.

- Error messages. The text in an error message might change, to make the wording clearer or more accurate.

Coexistence

Queue managers, with different names, can coexist on any server as long as they use the same IBM MQ installation. On z/OS, UNIX, Linux, and Windows, different queue managers can coexist on the same server and be associated with different installations. In addition to queue managers coexisting on a server, objects, and commands must work correctly with different queue managers running at different command levels.

z/OS

Multiple queue manager versions in z/OS

There can be several IBM MQ subsystems in a z/OS image, and they can use different versions of IBM MQ, provided that the IBM MQ early code modules are of the latest version being used. (These modules are loaded at z/OS IPL time and are shared among all the IBM MQ subsystems in the z/OS image.)

This means that you can run one queue manager at the latest version and another in the same image with an earlier version, provided that the early code is that of the latest version.

The coexistence section lists restrictions in the use of objects and commands when they are used with queue managers at multiple command levels. The queue managers might be running on a single server, or in a cluster.

Related concepts:

"Queue manager coexistence"
Queue managers, with different names, can coexist on any server as long as they use the same IBM MQ installation. On z/OS, UNIX, Linux, and Windows, different queue managers can coexist on the same server and be associated with different installations.

"Multi-installation queue manager coexistence on UNIX, Linux, and Windows" on page 668
You can install multiple copies of IBM MQ for UNIX, Linux, and Windows on the same server. The installations must be at Version 7.1 or later, with one exception. One Version 7.0.1 installation, at fix pack level 6, or later, can coexist with multiple Version 7.1, or later installations.

Related tasks:

"Migrating IBM MQ library loading from an earlier version of the product to the latest version" on page 684
No change in the way IBM MQ libraries are loaded is normally required if you upgrade from an earlier version of the product to the latest version. You must have followed the instructions on building IBM MQ applications in the earlier version, and you must replace the earlier version with the latest version of the product. If you choose to take advantage of multi-installation in the latest version of the product, based on the side-by-side or multi-stage migration scenarios, you must modify the environment for the operating system to resolve IBM MQ dependencies for an application. Typically, you can modify the runtime environment, rather than relink the application.

Queue manager coexistence: ULW

Queue managers, with different names, can coexist on any server as long as they use the same IBM MQ installation. On z/OS, UNIX, Linux, and Windows, different queue managers can coexist on the same server and be associated with different installations.

Single installation queue manager coexistence on all platforms

Single installation queue manager coexistence is useful in development and production environments. In development environments, you can set up different queue manager configurations to support different development activities. You can also work with multiple queue manager configurations on a single server, connected by channels, as if deployed on a network.

In production environments configuring multiple queue manager on a single server is less common. It has no performance or functional advantage over a single queue manager configuration. Sometimes, you must deploy multiple queue managers on server. It might be essential to meet the requirements of a particular software stack, governance, administration, or as a consequence of the consolidation of servers.

Queue manager coexistence in a multi-installation

ULW On UNIX, Linux, and Windows, multi-installation [3] queue manager coexistence became available in Version 7.1.

z/OS Multi-installation queue manager coexistence has always been supported on z/OS.

With multi-installation queue manager coexistence on the same server, you can run queue managers at different commands levels on the same server. You can also run multiple queue managers at the same command level, but associate them with different installations.

Multi-installation adds more flexibility to the coexistence of queue managers using a single installation. Any of the reasons behind running multiple queue managers, such as supporting different software stacks, might require different versions of IBM MQ.

The biggest benefit of multi-installation identified by early users, is in upgrading from one version of IBM MQ to another. Multi-installation makes upgrading less risky, less costly, and is more flexible in meeting the migration needs of applications running on a server.

The key to migration flexibility is being able to install a new version alongside an existing installation; see Figure 68 on page 666, which is extracted from “Migrating on UNIX and Linux: side-by side” on page 727 or “Migrating on Windows: side-by-side” on page 699.

3. Do not confuse multi-installation queue manager coexistence with multi-instance queue managers. They are completely different, though they sound similar in English.

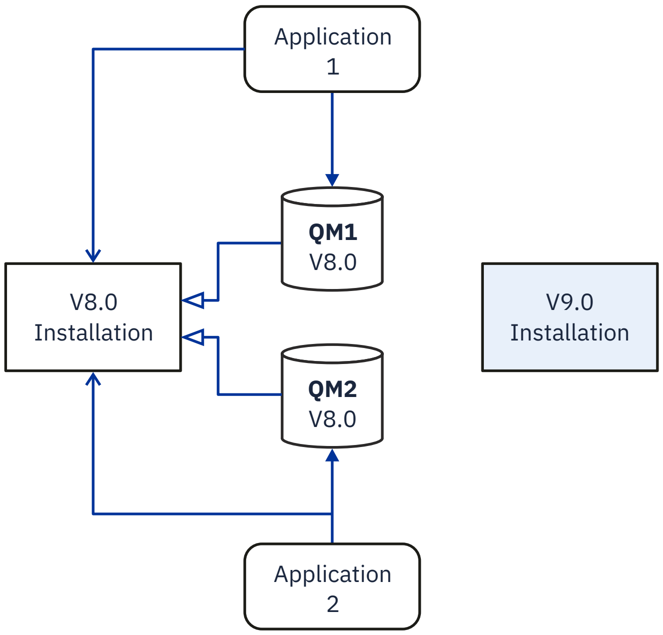


Figure 68. Side-by-side installation - step 2

When the installation is complete, and verified, migrate queue managers and applications to the new installation; see Figure 69. When migration is complete, uninstall the old installation.

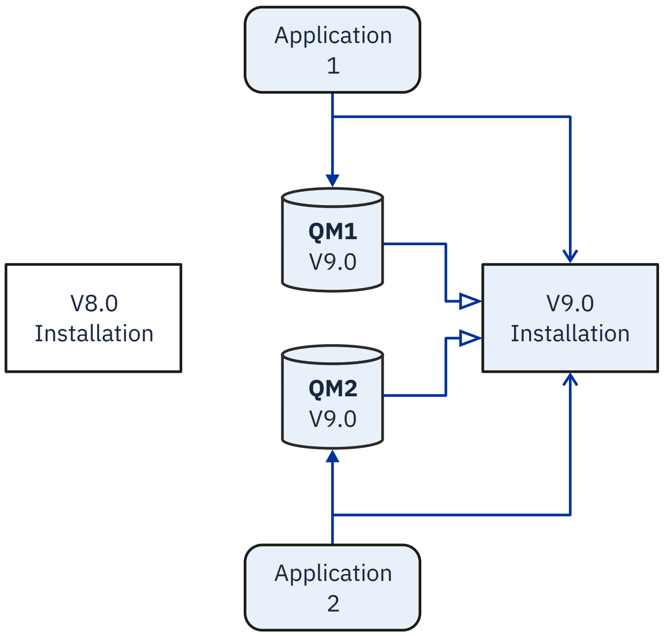


Figure 69. Side-by-side installation - step 4

Think of multi-installation as being the basis for a range of migration strategies. At one end is *single-stage*, in which you only have one installation on a server at a time. At the other end is *multi-stage* migration, in which you continue to run multiple installations at the same time. In the middle is side-by-side migration. Each of the three strategies is explained in the following tasks:

1. “Migrating on UNIX and Linux: single-stage” on page 724 or “Migrating on Windows: single stage” on page 694
2. “Migrating on UNIX and Linux: side-by side” on page 727 or “Migrating on Windows: side-by-side” on page 699
3. “Migrating on UNIX and Linux: multi-stage” on page 730 or or “Migrating on Windows: multi-stage” on page 701

LTS

Migration of queue managers to a new fix level

Another similar use of multi-installation is to support the migration of queue managers to a new fix level; see Figure 70. You maintain two installations, one of which has the latest fix pack applied, and the other has the previous maintenance levels. When you have moved all queue managers to the latest fix pack level, you can replace the previous fix pack with the next fix pack to be released. The configuration allows you to stage the migrating applications and queue managers to the latest fix pack level. You can switch the primary installation designation to the latest fix pack level.

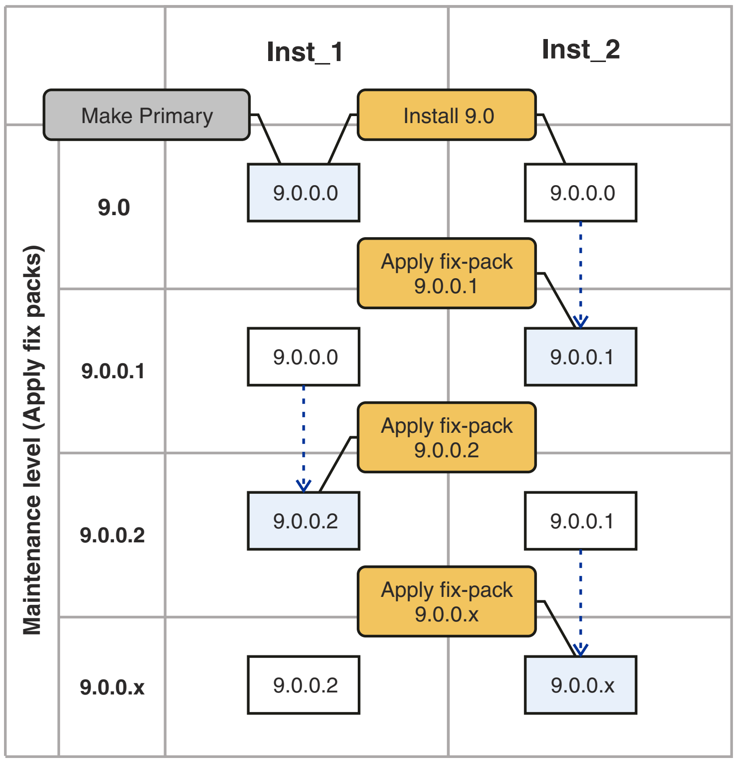


Figure 70. Rolling fix packs

Related concepts:

"Multi-installation queue manager coexistence on UNIX, Linux, and Windows"
You can install multiple copies of IBM MQ for UNIX, Linux, and Windows on the same server. The installations must be at Version 7.1 or later, with one exception. One Version 7.0.1 installation, at fix pack level 6, or later, can coexist with multiple Version 7.1, or later installations.

z/OS "Upgrade and migration of IBM MQ on z/OS" on page 785
You can install new releases of IBM MQ to upgrade IBM MQ to a new release, or version level. Multiple installations at the same or different levels can coexist on the same z/OS instance. Running a queue manager at a higher level requires migration.

Related tasks:

"Migrating IBM MQ library loading from an earlier version of the product to the latest version" on page 684
No change in the way IBM MQ libraries are loaded is normally required if you upgrade from an earlier version of the product to the latest version. You must have followed the instructions on building IBM MQ applications in the earlier version, and you must replace the earlier version with the latest version of the product. If you choose to take advantage of multi-installation in the latest version of the product, based on the side-by-side or multi-stage migration scenarios, you must modify the environment for the operating system to resolve IBM MQ dependencies for an application. Typically, you can modify the runtime environment, rather than relink the application.

"Migrating IBM MQ library loading to a later version on UNIX and Linux" on page 739
On UNIX and Linux, no change in the way IBM MQ libraries are loaded is normally required if you upgrade from an earlier version of the product to a later version by replacing an earlier version of the product with the later version, based on the single stage scenario. However, if you choose to take advantage of multi-installation in the later version of the product, based on the side-by-side or multi-stage migration scenarios, you might have to configure the runtime environment differently, for the operating system to load the later version of the IBM MQ library.

"Staging maintenance fixes on Windows" on page 583
On Windows systems, you can use multiple installations of IBM MQ on the same server to control the release of maintenance fixes.

"Staging maintenance fixes on UNIX and Linux" on page 608
On UNIX and Linux, you can use multiple installations of IBM MQ on the same server to control the release of maintenance fixes.

"Migrating IBM MQ library loading to a later version on Windows" on page 709
On Windows, no change in the way IBM MQ libraries are loaded is normally required if you upgrade from an earlier version of the product to the later version by replacing an earlier version of the product with the later version, based on the single stage scenario. However, if you choose to take advantage of multi-installation in the later version of the product, based on the side-by-side or multi-stage migration scenarios, you might have to configure the runtime environment differently, for the operating system to load the later version of the IBM MQ library.

Multi-installation queue manager coexistence on UNIX, Linux, and Windows: ULW

You can install multiple copies of IBM MQ for UNIX, Linux, and Windows on the same server. The installations must be at Version 7.1 or later, with one exception. One Version 7.0.1 installation, at fix pack level 6, or later, can coexist with multiple Version 7.1, or later installations.

Figure 71 on page 669 shows two IBM MQ installations, two queue managers, and three applications. Applications 2 and 3 are connected to `QM2`, and application 1 is connected to `QM1`. Applications 1 and 3 load IBM MQ libraries from the `Inst_1` installation, and application 2 loads libraries from the Version 7.0.1 installation.

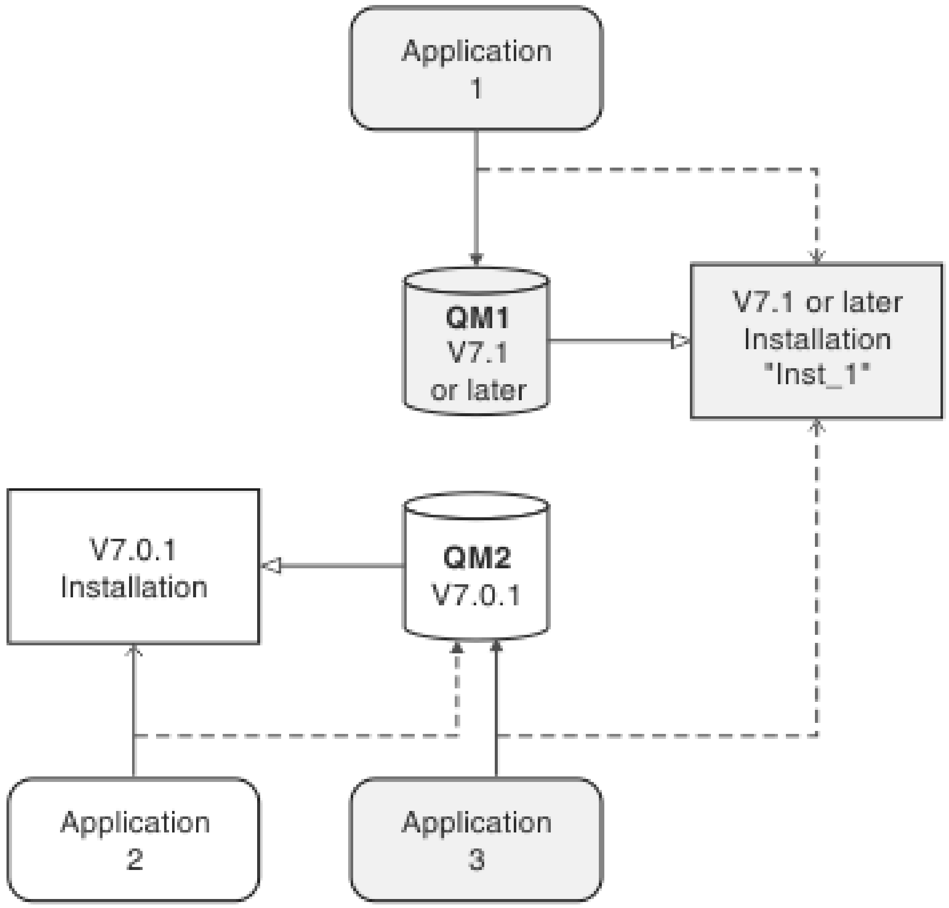


Figure 71. Coexistence of two queue managers using Version 7.0.1 and a later version installations

When you upgrade from Version 7.0.1 to a later version, you can choose to run Version 7.0.1 alongside the later version. The installation, illustrated in Figure 71, is called a multi-version installation. You can also install multiple copies of Version 7.1 alongside each other. That would be called multi-installation. Multi-installation is the more general term.

Version 7.0.1 did not support multi-installation on Multiplatforms. Before Version 7.1 becoming available, Version 7.0.1, Fix Pack 6 was shipped with some fixes to make Version 7.0.1 compatible with a later version on the same server. With Version 7.0.1, Fix Pack 6 installed, you can run one copy of Version 7.0.1 alongside multiple copies of the later version. You do not have to apply the fix pack to upgrade Version 7.0.1 to Version 7.1 “in-place” ; see “Migrating on UNIX and Linux: single-stage” on page 724 or “Migrating on Windows: single stage” on page 694.

A multi-version installation that includes Version 7.0.1, does not behave the same way as a multi-installation that does not. The differences primarily affect how you might choose to configure how applications load IBM MQ libraries, and run IBM MQ commands. Because of these differences, think of the multi-version support provided in Version 7.0.1, Fix Pack 6, as a migration aid to moving to a later version multi-installation environment. The topics that explain the restrictions in Version 7.0.1 multi-version are listed in related links.

If you run multiple installations of IBM MQ on a server you must consider three questions:

1. Which installation is a queue manager associated with; see “Queue manager association” on page 670?

2. Which installation does an application load; see "Loading IBM MQ libraries"?
3. Which installation is an IBM MQ command run from; see "Command association" on page 672?

Queue manager association

Before Version 7.1, queue managers on UNIX, Linux, or Windows were associated with the only installation on the server. With Version 7.1, or later, installed on the same server as Version 7.0.1, you can change the association of a queue manager to a later version by running **`setmqm`** ; see setmqm. You cannot change the association of a queue manager running a release of IBM MQ earlier than Version 7.0.1 because you cannot install a later version of the product on a server with an installation of IBM MQ earlier than Version 7.0.1.

A queue manager is permanently associated with an installation, until you choose to change the association with the **`setmqm`** command. You cannot associate a queue manager with an installation at a lower command level than the current command level of the queue manager.

In Figure 71 on page 669, `QM1` is associated with `Inst_1`. The association is made by running `setmqm -m QM1 -n Inst_1`. When `QM1` is first started, after running **`setmqm`**, if `QM1` was running Version 7.0.1, it is migrated to a later version. `QM2` is associated with Version 7.0.1 because the association has not been changed.

Loading IBM MQ libraries

The application connections to the queue managers are established by calling MQCONN or MQCONNX in the normal way.

Which IBM MQ library an application loads depends on the configuration of the operating system loader and on the IBM MQ installation the queue manager is associated with.

In Figure 71 on page 669, the operating system loads the IBM MQ library from the `Inst_1` installation for applications 1 and 3. It loads the IBM WebSphere MQ Version 7.0.1 library for application 2. The operating system has loaded the wrong library for application 3. Application 3 requires the IBM WebSphere MQ Version 7.0.1 libraries.

Figure 72 on page 671 shows what happens to application 3. Application 3 is connecting to `QM2`, and `QM2` is associated with the IBM WebSphere MQ Version 7.0.1 installation. IBM MQ detects that the operating system has loaded the wrong library to process calls from application 3 to `QM2`. IBM MQ loads the correct library from the IBM WebSphere MQ Version 7.0.1 installation. It transfers the MQCONN or MQCONNX call to the IBM WebSphere MQ Version 7.0.1 library. Subsequent MQI calls that use the connection handle returned by MQCONN or MQCONNX, call entry points in the IBM WebSphere MQ Version 7.0.1 library.

Because IBM WebSphere MQ Version 7.0.1 libraries cannot load IBM MQ libraries from other installations, there is no corresponding application in Figure 72 on page 671 that loads a IBM WebSphere MQ Version 7.0.1 library and connects to a queue manager running Version 7.1. If you attempt a connection to `QM1` with application 2, IBM MQ returns an error; see 2059 (080B) (RC2059): MQRC_Q_MGR_NOT_AVAILABLE.

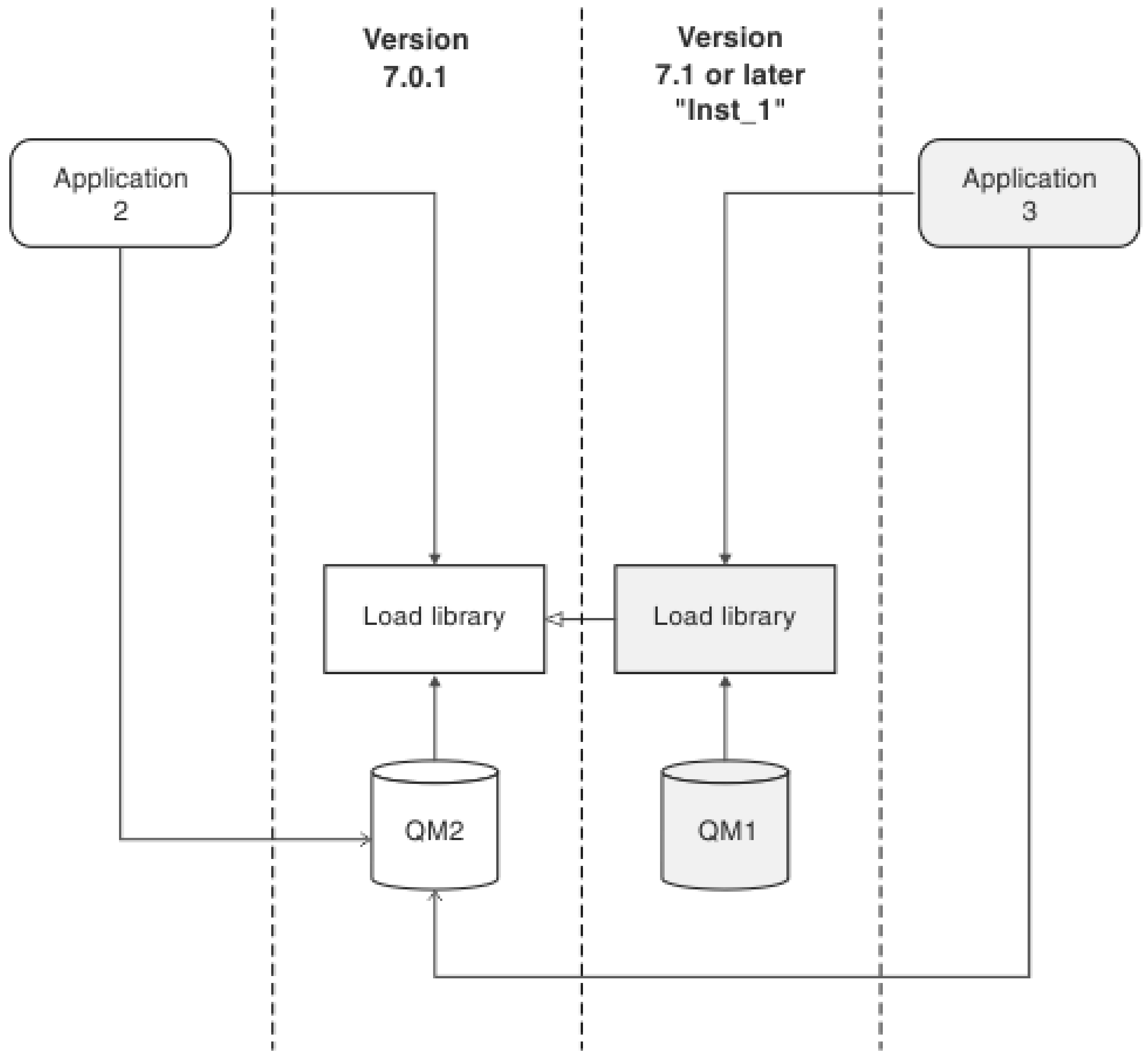


Figure 72. Loading calls in a different library

A Version 7.1, or later, IBM MQ library includes a routing capability that is based on the installation a queue manager is associated with. Earlier IBM MQ libraries do not have a routing capability. The operating system can load a library from any Version 7.1 installation, or later, and IBM MQ transfers MQI calls to the correct library.

The new loading capability of IBM MQ libraries in Version 7.1, or later, does not relax the restriction that an application compiled and linked at a later release level must not directly load an IBM MQ library at an earlier release level. In practice the restriction is of less significance than in earlier releases, because as long as the operating system loads a library at the same or later level than the library the application was compiled and linked with, IBM MQ can call any other level of IBM MQ on the same server from Version 7.0.1 upwards.

For example, suppose you recompile and link an application that is to connect to a Version 7.0.1 queue manager using the libraries shipped with Version 7.1. At run time the operating system must load the Version 7.1 libraries for the application, even though the application connects to a Version 7.0.1 queue manager. IBM WebSphere MQ Version 7.1 detects the inconsistency and loads the Version 7.0.1 library for the application. The same applies to any future release. If the application is recompiled and linked against a later release, then the application must load an IBM MQ library that matches the later release, even if it continues to connect to a Version 7.1 queue manager.

Your application might not be linked to an IBM MQ library, but instead calls the operating system directly to load an IBM MQ library. If the library that is loaded is from Version 7.1 or later, IBM MQ checks the library is from the installation that is associated with the queue manager. If it is not, IBM MQ loads the correct library.

Special migration considerations involving loading IBM MQ libraries

You might have been asked to modify the installation of an earlier IBM MQ release to satisfy the requirements of a build environment, or the IT standards in your organization. If you copied IBM MQ libraries to other directories, or created symbolic links, you ended up with an unsupported configuration. The requirement to move IBM MQ libraries to other directories was one of the reasons for changing the installation of IBM MQ on UNIX and Linux. You can now install IBM MQ into a directory of your choosing. You can also load IBM MQ libraries from the `/usr/lib` directory, which is normally on the default load path on UNIX and Linux systems.

A common IT standard or build environment requirement is to include IBM MQ libraries in the default load path on UNIX and Linux systems. IBM WebSphere MQ Version 7.1 has a solution. In Version 9.0 you can install IBM MQ into a directory of your own choosing, and IBM MQ can create symbolic links in `/usr` and its subdirectories. If you make a Version 7.1, or later, installation primary by using the **`setmqinst`** command, IBM MQ inserts symbolic links to the IBM MQ libraries into `/usr/lib`. As a result, the operating system finds the IBM MQ libraries in the default load path, if that includes `/usr/lib`.

Because IBM WebSphere MQ Version 7.1, or later, libraries transfer calls to the correct installation, defining that version installation as primary, also results in the correct libraries being loaded for any application that is built with a link to `/usr/lib`, regardless of which queue manager it connects to. Unfortunately, this solution does not work if you have a Version 7.0.1 installation on the server, because then you cannot define a Version 7.1, or later, installation as primary, and the Version 7.0.1 libraries do not load libraries from other installations. As an alternative to setting the later version installation primary, use **`setmqenv`** with the `-k` or `-l` options to achieve a similar result.

You can find more information in Connecting applications in a multiple installation environment.

Command association

Examples of commands are **`dspmqver`**, **`setmqinst`**, **`runmqsc`**, and **`strmqm`**. The operating system must find a command in an IBM MQ installation. Many commands also require a queue manager as an argument and assume the default queue manager, if a queue manager name is not provided as a parameter.

Unlike loading libraries, if a command includes a queue manager as a parameter, the command is not switched to the installation that is associated with the queue manager. You must use the **`setmqenv`** command to set up your environment correctly, so that any commands that you issue are run from the correct installation. You can provide a queue manager as a parameter to **`setmqenv`**, to set up the command environment for that queue manager; see Figure 73 on page 673.

On Windows, the **`setmqinst`** command sets global environment variables, and **`setmqenv`** local environment variables, including the `PATH` variable to find commands.

On UNIX and Linux, the **`setmqinst`** command copies symbolic links for a subset of the commands into `/usr/bin` ; see External library and control command links to primary installation on UNIX and Linux. The **`setmqenv`** command sets up local environment variables, including the search path to the binary folder in the installation directory.

setmqenv must be on the search path in order to run it. One reason for having a later version installation as primary is to be able to run setmqenv without having to configure the search path. If IBM WebSphere MQ Version 7.0.1 is installed on the server, no Version 7.1, or later, installation can be primary and IBM WebSphere MQ Version 7.0.1 does not have a setmqenv command. The consequence is, you must provide a path to run the setmqenv command to set up the command environment for any of the later version installations on the server.

Figure 73 shows two examples of running **`setmqenv`** to set up the command environment for the copy of IBM MQ that is associated with the queue manager, `QM1`.

IBM WebSphere MQ for Windows Version 7.1

```
" MQ_INSTALLATION_PATH\bin\setmqenv" -m QM1
```

IBM WebSphere MQ Version 7.1 for UNIX and Linux

```
. MQ_INSTALLATION_PATH/bin/setmqenv -m QM1
```

Figure 73. Running **`setmqenv`**

Related tasks:

"Migrating IBM MQ library loading from an earlier version of the product to the latest version" on page 684
No change in the way IBM MQ libraries are loaded is normally required if you upgrade from an earlier version of the product to the latest version. You must have followed the instructions on building IBM MQ applications in the earlier version, and you must replace the earlier version with the latest version of the product. If you choose to take advantage of multi-installation in the latest version of the product, based on the side-by-side or multi-stage migration scenarios, you must modify the environment for the operating system to resolve IBM MQ dependencies for an application. Typically, you can modify the runtime environment, rather than relink the application.

"Staging maintenance fixes on Windows" on page 583
On Windows systems, you can use multiple installations of IBM MQ on the same server to control the release of maintenance fixes.

"Staging maintenance fixes on UNIX and Linux" on page 608
On UNIX and Linux, you can use multiple installations of IBM MQ on the same server to control the release of maintenance fixes.

"Migrating IBM MQ library loading to a later version on UNIX and Linux" on page 739
On UNIX and Linux, no change in the way IBM MQ libraries are loaded is normally required if you upgrade from an earlier version of the product to a later version by replacing an earlier version of the product with the later version, based on the single stage scenario. However, if you choose to take advantage of multi-installation in the later version of the product, based on the side-by-side or multi-stage migration scenarios, you might have to configure the runtime environment differently, for the operating system to load the later version of the IBM MQ library.

"Migrating IBM MQ library loading to a later version on Windows" on page 709
On Windows, no change in the way IBM MQ libraries are loaded is normally required if you upgrade from an earlier version of the product to the later version by replacing an earlier version of the product with the later version, based on the single stage scenario. However, if you choose to take advantage of multi-installation in the later version of the product, based on the side-by-side or multi-stage migration scenarios, you might have to configure the runtime environment differently, for the operating system to load the later version of the IBM MQ library.

Related reference:

"Coexistence" on page 664
Queue managers, with different names, can coexist on any server as long as they use the same IBM MQ installation. On z/OS, UNIX, Linux, and Windows, different queue managers can coexist on the same server and be associated with different installations. In addition to queue managers coexisting on a server, objects, and commands must work correctly with different queue managers running at different command levels.

Related information:

Changing the primary installation

Connecting applications in a multiple installation environment

Dynamic-Link Library Search Order

External library and control command links to primary installation on UNIX and Linux

Features that can be used only with the primary installation on Windows

Installation configuration file, mqinst.ini

setmqenv

setmqinst

setmqm

`strmqm` -e CMDLEVEL

JMS: Administered objects:

Version 6.0 JMS administered objects, such as connection factories and destinations are compatible with later releases.

JMS applications can use connection factory or destination objects created in IBM WebSphere MQ Version 6.0. Any new properties, that did not exist in Version 6.0, assume their default value.

IBM WebSphere MQ Version 6.0 JMS applications can use connection factory or destination objects created in later releases. Any of the new properties that do not exist in Version 6.0 are ignored.

Mixed version cluster coexistence:

A cluster can contain queue managers running at IBM MQ Version 9.0, and any currently supported earlier level of the product. However new features cannot be exploited from queue managers at an earlier level.

Cluster workload balancing in a mixed version cluster

IBM MQ Version 7.1 added a new **DEFBIND** value called `GROUP` to queues. Applications on queue managers earlier than Version 7.1 must not open or put messages to queues specifying the new value. When an application ignores this limitation, the workload balancing behavior (for example: BIND_ON_OPEN or BIND_NOT_FIXED) is undefined.

Routing behavior in a mixed version publish/subscribe cluster

From IBM MQ Version 8.0, topic host routing is available for publish/subscribe clusters. The queue manager where the object is defined, and the full repository queue managers, must be at a level that supports the topic route hosting feature, that is Version 8.0 or later. Any queue manager in the cluster that is at an earlier level does not adhere to the topic route hosting behavior.

When a clustered topic is defined for topic host routing (by setting the topic **CLROUTE** parameter to `TOPICHOST`), only queue managers at the new level are aware of the clustered topic. Older queue managers do not receive the clustered topic definition and therefore behave as if the topic is not clustered. This means that all queue managers that need to work in a routed publish/subscribe manner must be at a version that supports this feature, not just the queue managers that host the routed topics.

Important notes:

- All full repositories must be at Version 8.0 or later to use this feature. If a full repository queue manager is at an earlier version, the **`CLROUTE`** of `TOPICHOST` is not recognized by the full repository, and the full repository propagates the topic definition to all queue managers in the cluster. Any pre-Version 8.0 queue managers then use the topic as if it is defined for `DIRECT` routing. This behavior is unsupported.
- If an older queue manager defines a direct routed clustered topic with the same name as an existing topic host routed clustered topic, the full repositories notice the conflicting definition and do not propagate the definition.

To find out the version of each queue manager in the cluster, specify the VERSION parameter with the DISPLAY CLUSQMGR command. If you issue this command from a queue manager with a full repository, the information returned applies to every queue manager in the cluster. Otherwise the information returned applies only to the queue managers in which it has an interest. That is, every queue manager to which it has tried to send a message and every queue manager that holds a full repository.

ISPF operations and control panels on z/OS: z/OS

When using the operations and control panels, the IBM MQ libraries you use in ISPF must be compatible with the queue manager you are working with.

Table 82 shows which versions of the operations and controls panels you use in ISPF are compatible with which levels of queue manager. Version 7.1, or later, panels are incompatible with any release before Version 6.0. CD

Table 82. Compatibility of queue manager versions with operations and control panel versions on z/OS

Version	Queue-sharing group containing a mixture of Version 7.1 Version 8.0, and Version 9.0 queue managers.	Version 9.0 queue manager	Version 8.0 queue manager	Version 7.1 queue manager
Version 9.0 panel	Compatible with restrictions and warnings	Compatible	Compatible	Compatible with restrictions and warnings
Version 8.0 panel	Not compatible	Not compatible	Compatible	Compatible with restrictions and warnings
Version 7.1 panel	Not compatible			Compatible

Queue-sharing group coexistence on z/OS: z/OS

A queue-sharing group can contain queue managers running on IBM WebSphere MQ Version 7.1.0, and on later releases. The queue managers can access the same shared queues and other shared objects. Queue managers running earlier versions of the product must have the coexistence PTF applied for the latest release.

Notes:

1. After the coexistence PTF has been applied, the earlier version queue managers must be started at least once.
2. Long Term Support release and Continuous Delivery release queue managers, with the same version and release numbers, can coexist in a queue-sharing group without the need for a coexistence PTF.

Only run queue managers in a mixed-version queue-sharing group for the time it takes to migrate all queue managers to the later version. If the queue-sharing group contains queue managers with a mixture of versions, new functions on the latest version, that are restricted by OPMODE, will not be available.

Properties of objects in a mixed queue-sharing group on z/OS: z/OS

Attributes that did not exist in earlier versions can be created and altered on queue managers of a later version in a mixed queue-sharing group. The attributes are not available to queue managers in the group that are at an earlier level.

Any **`QSGDISP`**`(GROUP) TOPIC` objects having the **`CLROUTE`**`(TOPICHOST)` attribute set, and any **`QSGDISP`**`(GROUP) AUTHINFO` objects with **`AUTHTYPE`**`(IDPWOS)`, are hidden from queue managers earlier than Version 8.0 in a mixed queue-sharing group.

MQSC commands in a mixed queue-sharing group on z/OS: z/OS

Existing **`MQSC`** commands using new keywords and attribute values can be entered for routing to a migrated queue manager. You can enter the commands on any queue manager. Route the commands using **`CMDSCOPE`**. Commands with new keywords and attribute values, or new commands, routed to a previous version of queue manager, fail.

Application compatibility and interoperability with earlier versions of IBM MQ

Connecting an application that is built against libraries shipped with a later version of IBM MQ to an earlier version IBM MQ is not supported. Avoid building applications against a later version, and redeploying them to a queue manager running at an earlier version, although some applications do work in practice.

IBM MQ applications do interoperate with applications running on earlier versions of IBM MQ, as long as they use no new function. IBM MQ clients can connect to queue managers running at an earlier version than the client, as long as the client uses no new functions.

An IBM MQ application that uses only functions provided by an earlier version of a queue manager can continue to send messages to the earlier version. It does not matter what version of IBM MQ an application is built on and connected to. It can exchange messages with an application connected to an earlier version of IBM MQ, as long as it does not use new function.

Consider these four cases; the first two cases are not supported though they might work in practice, the last two cases are supported. The first two cases require compatibility with an earlier version of IBM MQ. The last two cases rely on the interoperability between all versions of IBM MQ

1. Running an IBM MQ server application, built with a later version of IBM MQ, connecting to a queue manager running on a server with an earlier version of IBM MQ installed.

2. Running an IBM MQ client application, built with a later version of IBM MQ, on a client platform with an earlier client installation, connecting to a queue manager running on a server with a later version of IBM MQ installed.
3. Running an IBM MQ client application, built with a later version of IBM MQ, on a client platform with the later client installation, connecting to a queue manager running on a server with an earlier version of IBM MQ installed.
4. Exchanging messages between an IBM MQ client or server application, connected to a queue manager running on a server with a later version of IBM MQ installed, with applications connected to a queue manager running on a server with an earlier version of IBM MQ installed.

Plan to avoid the first two cases, as they are not guaranteed to work all the time. If you are running an incompatible configuration and you encounter a problem, you must rebuild your applications with the correct level of IBM MQ. You can then continue with problem diagnosis.

Multi-installation and application loading

The new loading capability of IBM MQ libraries in Version 7.1, or later, does not relax the restriction that an application compiled and linked at a later release level must not directly load an IBM MQ library at an earlier release level. In practice the restriction is of less significance than in earlier releases, because as long as the operating system loads a library at the same or later level than the library the application was compiled and linked with, IBM MQ can call any other level of IBM MQ on the same server from Version 7.0.1 upwards.

For example, suppose you recompile and link an application that is to connect to a Version 7.0.1 queue manager using the libraries shipped with Version 7.1. At run time the operating system must load the Version 7.1 libraries for the application, even though the application connects to a Version 7.0.1 queue manager. IBM WebSphere MQ Version 7.1 detects the inconsistency and loads the Version 7.0.1 library for the application. The same applies to any future release. If the application is recompiled and linked against a later release, then the application must load an IBM MQ library that matches the later release, even if it continues to connect to a Version 7.1 queue manager.

Examples

1. You decide to rebuild a client application. Can you deploy it to your production environment that contains some earlier versions of client and server platforms?

   The answer is no, you must upgrade all the client workstations you deploy to, at least to the version of the client you have built. The queue managers running on earlier versions of IBM MQ do not have to be upgraded. In practice all the clients are likely to work, but for maintainability you must avoid running incompatible levels of an application and the IBM MQ client.
2. You deploy some IBM MQ queue managers at a new version level. You have an existing IBM MQ application that you use to send messages between the servers. Do you rebuild the application to deploy it onto the new servers? Can you deploy the old version onto the new servers?

   The answer is, either. You can continue to deploy the existing version of the application onto all your servers, or you can deploy the rebuilt application onto the new servers. Either configuration works. IBM MQ supports running the existing application on later servers and sending messages from later application versions to earlier ones. What you must not do is to rebuild the application on the later version and redeploy it onto both the earlier and newer servers. IBM MQ does not support compatibility with earlier versions.

z/OS

z/OS application stubs

The stub modules that are listed are link-edited with applications and exits. The version 7 stub modules might not work with Version 6.
- `CSQASTUB`

- CSQBRSSI
- CSQBRSTB
- CSQBSTUB
- CSQCSTUB
- CSQQSTUB
- CSQXSTUB

To take advantage of the new APIs introduced in Version 7.0, for example MQSUB and MQCB, you must link-edit your application with stubs shipped with Version 7.0 or later, for an LE program, the sidedeck, see Building z/OS batch applications using Language Environment . Such an application will not run on an earlier version of the queue manager.

Application compatibility and interoperability with later versions of IBM MQ

IBM MQ applications run against later versions of a queue manager without recoding, recompiling, or relinking. You can connect an application that is built against libraries shipped with an earlier version of IBM MQ to a queue manager running at a later version of IBM MQ.

If you upgrade a queue manager to a later version, existing applications built against its earlier version work without change. Exceptions are noted in “Changes that affect migration” on page 626. Likewise applications connected to the IBM MQ Client, run against later versions of the client without recoding, recompiling, or relinking. You can deploy client applications built against earlier versions of the IBM MQ Client libraries to connect using later versions of the libraries.

All the following four cases are supported. The first two cases rely on the compatibility of later version of IBM MQ with applications built against earlier versions. The last two cases rely on the interoperability between all versions of IBM MQ.

You might change the operating environment as a prerequisite of migrating to a new level of queue manager. The operating environment changes, rather than changes in IBM MQ itself, might require application change, recompilation, or relinking. Sometime the operating environment change affects only the development environment, and the operating environment supports applications built at an earlier level. In which case, you might be able to run existing applications built at the older level of the operating environment. You might not be able to build any new applications until the operating environment is upgraded.

In the future, after you have migrated queue managers and clients to the latest release level, consider changing your applications to take advantage of new capabilities.

z/OS

z/OS application stubs

The stub modules that are listed are link-edited with applications and exits. The Version 6.0 stub modules continue to work with Version 9.0.
- CSQASTUB
- CSQBRSSI
- CSQBRSTB
- CSQBSTUB
- CSQCSTUB
- CSQQSTUB
- CSQXSTUB

To take advantage of the new APIs introduced in Version 7.0, for example MQSUB and MQCB, you must link-edit your application with stubs shipped with Version 7.0 or later, for an LE program, the sidedeck, see Building z/OS batch applications using Language Environment . Such an application will not run on an earlier version of the queue manager.

Compatibility between different versions of an IBM MQ MQI client and a queue manager

Any version and release of an IBM MQ MQI client can connect to any version and release of an IBM MQ queue manager. The MQI channel is automatically configured to the latest version that both the client and server support. If the client and server are different versions, the client application must use only the functions in the earlier version.

The compatibility between clients and queue managers applies only to the version and release (`V.R`) of the product. The statement of compatibility does not necessarily apply to the modification and fix pack level (`M.F`) of the product.

If there are known problems at a specific `V.R.M.F` of the product, an upgrade to a more recent fix pack for the same `Version.Release` is necessary.

When you upgrade a queue manager to a different version, you automatically upgrade IBM MQ libraries. The libraries are used by IBM MQ MQI client and server applications running on the same server as the queue manager. To access new functions from remote clients, you must also upgrade the IBM MQ MQI client installation on remote workstations. The IBM MQ MQI client includes the IBM MQ MQI client libraries.

Remote clients that have not been upgraded continue to work with an upgraded queue manager. The behavior of the client application might, in rare cases change. You must consult "Changes that affect migration" on page 626, to find out whether changes in the current version affect your client applications.

Remote clients that are connected to upgraded queue managers can use the new functions in the release. If an upgraded remote client is connected to a queue manager that has not been upgraded, it must not use new functions. In rare cases, the behavior of the client might change; see "Changes that affect migration" on page 626.

You can generally assume that upgrading the IBM MQ MQI client does not require you to recompile or relink the client application. You can also continue to use the same connection to the queue manager. If changes are required, they are identified in "Migrating a queue manager on Windows" on page 691, for the particular migration path and platform you are concerned with.

The Client Channel Definition Table (CCDT) is an interface to customize the connection between an IBM MQ Client and a queue manager. Entries in the tables are client connections, which are defined using a queue manager. The version of a CCDT is the version of the queue manager used to define the client connections. If an IBM MQ MQI client uses CCDT to connect to a queue manager, the CCDT can be at a version greater than, less than, or equal to that of the client.

You can connect to a queue manager with an earlier IBM MQ Client or an earlier CCDT. If you are using a CCDT, and you plan to use new client channel configuration options, such as shared conversations, you must upgrade the CCDT, and therefore the IBM MQ MQI client installation to the new version.

MQI client: Client Channel Definition Table (CCDT):

You can connect an IBM MQ MQI client application to any level of queue manager. If a client uses CCDT to connect to a queue manager, the CCDT can be at a version greater than, less than, or equal to that of the client.

Version of originating queue manager for a CCDT

In earlier releases before IBM MQ Version 9.0, clients can use a CCDT built by the same or earlier version queue manager, but there was previously a restriction on clients using a CCDT built by a later version queue manager. However, this restriction is removed in Version 9.0.

From Version 9.0, if a client uses a CCDT, it can use a CCDT built by a later version queue manager, as well as a CCDT built by the same, or earlier version of queue manager.

The same restriction on the use of CCDTs originating from later version queue managers is also removed in Version 8.0, Version 7.5, and Version 7.1 by APARs IT10863 and IT11547. for more information, see the technote MQ 7.x, MQ 8.0 and MQ 9.0 compatibility with previous versions - including usage of CCDT files, JMS .bindings, SSL/TLS.

Common migration scenarios

If, for example, you upgrade a queue manager from an earlier release to a later release, and you do not create new CCDTs for its clients, the clients connect to the later release queue manager without any changes being required. Client behavior might change as a result of changes to the queue manager.

Another common migration scenario is to update some queue managers and some clients to a later release, leaving other queue managers and clients at the earlier release. In this scenario, you want to update the CCDT for the IBM MQ MQI clients that are connected to the later release queue managers to that later release so that those clients can fully use the function in the later release. The new clients can also connect to the earlier release queue managers. Existing clients connect to queue managers in both releases. In order that the clients in the later release can use the new function in that release, you must deploy a CCDT that has been generated by a queue manager in that new release. Clients in the earlier release can continue to use the CCDT for that earlier release. Both sets of clients can connect to both sets of queue managers, regardless of the CCDT they are using.

IBM MQ MQI clients

If the client is an IBM MQ MQI client, the version of the IBM MQ MQI client libraries that are linked to by the client must be the same or greater than the version of the queue manager that was used to build the CCDT.

To upgrade an IBM MQ MQI client from an earlier release to use a CCDT for a later release, you must upgrade the IBM MQ MQI client installation to the later release. Unless you decide to do so for other reasons, do not rebuild the client application.

Java or JMS client

If the client is a Java or JMS client, then the client must be built with versions of the IBM MQ JAR files that are the same or greater than the queue manager that was used to build the CCDT.

To upgrade a Java or JMS client from an earlier release to use a CCDT for a later release, redeploy the IBM MQ JAR files to the client workstation. You do not need to rebuild the Java or JMS client with the new JAR files.

Related information:

Client channel definition table

V 9.0.0 Web addressable access to the client channel definition table

Accessing client-connection channel definitions

MQI client: Client configuration stanzas moved into a new configuration file:

Client configuration information is moved from existing configuration stanzas into a new configuration file, `mqclient.ini`.

Moving client configuration information affects existing settings; for example:

- Set the TCP `KeepAlive` attribute for client connections in `mqclient.ini` ; for example:

  ```
  TCP:
  KeepAlive = Yes
  ```

  An existing setting in `qm.ini` is ignored.
- Set `ClientExitPath` in `mqclient.ini` ; for example:

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

  An existing setting in `mqs.ini` is moved to the client configuration file when you upgrade the client. If you add values to `mqs.ini`, they are ignored.
- Set `JavaExitsClasspath` in `mqclient.ini`.

  Do not continue to use the Java system property `com.ibm.mq.exitClasspath`. Existing settings continue to work, but they are deprecated. The setting in `mqclient.ini` has precedence over the Java system property.

Related information:

The IBM MQ classes for JMS configuration file

Assigning channel exits for IBM MQ classes for JMS

IBM MQ client configuration file

MQI client: Default behavior of client-connection and server-connection channels:

In Version 7.0 the default settings for client and server connection channels changed to use shared conversations. This change affects the behavior of heartbeats and channels exits, and can have an impact on performance.

Before Version 7.0, each conversation was allocated to a different channel instance. From Version 7.0, the default for client and server connections is to share an MQI channel. You use the **SHARECNV** (sharing conversations) parameter to specify the maximum number of conversations that can be shared over a particular TCP/IP client channel instance. The possible values are as follows:

SHARECNV(0)
: This value specifies no sharing of conversations over a TCP/IP socket. The channel instance behaves exactly as if it was a Version 6.0 server or client connection channel, and you do not get the extra features such as bi-directional heartbeats that are available when you set **SHARECNV** to 1 or greater. Only use a value of `0` if you have existing client applications that do not run correctly when you set **SHARECNV** to 1 or greater.

SHARECNV(1)
: This value specifies no sharing of conversations over a TCP/IP socket. Performance on distributed

servers is similar to that for a value of 0. Client heartbeating (whether in an MQGET call or not) and read ahead are available, and channel quiescing is more controllable. You can usually use this setting with existing Version 6.0 client applications.

SHARECNV(2) to SHARECNV(999999999)
Each of these values specifies the number of shared conversations. If the client-connection **SHARECNV** value does not match the server-connection **SHARECNV** value, then the lowest value is used. The default value is SHARECNV(10), which specifies 10 threads to run up to 10 client conversations per channel instance. However, on distributed servers there are performance issues with SHARECNV channels that can be eased by using SHARECNV(1) wherever possible.

For all **SHARECNV** values of 1 or greater, the channel supports the following features:
- Bi-directional heartbeats
- Administrator stop-quiesce
- Read-ahead
- Asynchronous-consume by client applications

You can also set the MQCONNX option, MQCNO_NO_CONV_SHARING and connect the application to a channel with **SHARECNV** set to a value greater than 1. The result is the same as connecting the application to a channel with **SHARECNV** set to 1.

Performance

The change in Version 7.0 to use shared conversations, and further enhancements introduced in Version 8.0, can impact performance on distributed servers. See Tuning client and server connection channels.

Heartbeats

From Version 7.0, heartbeats can flow across the channel at any time in either direction. The SHARECNV(0) and Version 6.0 behavior is for heartbeats to flow only when an MQGET call is waiting.

Channel exits

The behavior of a client or server connection channel exit changes when the channel is sharing conversations (that is, when you set **SHARECNV** to a value greater than 1). It is unlikely, but possible, that the change affects the behavior of existing exits. The change is as follows:
- Send or receive exits can alter the MQCD structure on an MQXR_INIT call. The effect of these exits differs, depending on whether the conversation is shared with other conversations on the same channel:
  - If the MQCXP SharingConversations field passed to the exit instance is set to FALSE, this exit instance is the first, or only, conversation on the channel instance. No other exit can be changing the MQCD at the same time, and changes that are made to the MQCD can affect the way that the channel runs.
  - If the MQCXP SharingConversations field passed to the exit instance is set to TRUE, this exit instance is a subsequent conversation. It is sharing the channel instance with other conversations. Changes made to the MQCD in the exit instance are retained in the MQCD but do not affect the way the channel runs.
- Send, receive, and security exit instances can alter the MQCD, when the MQCXP SharingConversations field is set to TRUE. Exit instances on other conversations might be changing the MQCD at the same time. Updates written by one exit instance can be overwritten by another instance. It might be necessary to serialize access to the MQCD across these different exit instances to maintain the consistency of fields in MQCD.

Updating MQCD when the SharingConversations field is set to TRUE does not affect the way the channel runs. Only alterations made when the MQCXP SharingConversations field is set to FALSE, on an MQXR_INIT call, change channel behavior.

Related information:
Using sharing conversations
Channel-exit programs for MQI channels
Using read ahead
Stopping MQI channels
Tuning client and server connection channels
HeartbeatInterval (MQLONG)
SharingConversations (MQLONG)
ALTER CHANNEL
The Asynchronous consume sample program

MQI client: MQPUT1 sync point behavior change:

An MQPUT1 call by an IBM MQ MQI client application that failed in IBM WebSphere MQ Version 6.0 can now sometimes succeed. The failure is returned to the application later, if it calls MQCMIT. For the change in behavior to occur, the MQPUT1 must be in sync point.

In the scenario, "Example call sequence that demonstrates the change in behavior," an MQPUT1 call can succeed where it failed in Version 6.0. The result occurs when all the following conditions are met:
- Both client and queue manager are later than Version 6.0.
- The application program is connected to the queue manager as a client application
- `MQPMO_SYNCPOINT` is set in the Put Message Options structure, `MQPMO`

You can make the IBM MQ MQI client behave like Version 6.0. Set **`Put1DefaultAlwaysSync`** to `YES` in the **`CHANNELS`** stanza of the client configuration file; see Figure 74.

```
Channels:
Put1DefaultAlwaysSync=YES
```

Figure 74. Add **`Put1DefaultAlwaysSync`** *to `mqclient.ini`*

Example call sequence that demonstrates the change in behavior

1. MQCONN to queue manager from an IBM MQ MQI client application.
2. MQPUT1 to a nonexistent queue with the `MQPMO_SYNCPOINT` option
3. MQDISC

In IBM WebSphere MQ Version 6.0 the MQPUT1 call ends with `MQCC_FAILED` and `MQRC_UNKNOWN_OBJECT_NAME(2085)`. Running with a client and server later than Version 6.0, the MQPUT1 call ends with `MQRC_NONE` and `MQCC_OK`.

Related information:
CHANNELS stanza of the client configuration file

Overview of migration requirements

An overview of migration requirements, on all platforms, are grouped in this section.

This information has moved. See "Migration concepts and methods" on page 645.

Related concepts:
"Migration concepts and methods" on page 645
An overview of the various concepts and methods for migrating from one release of the product to another.

Migrating MQ Telemetry to the latest version

Migrate MQ Telemetry to the latest version of the product by completing the tasks in this section. You must stop all IBM MQ activity on the system before migrating.

About this task

This information has moved. See "Migrating MQ Telemetry on Windows" on page 716 and "Migrating MQ Telemetry on Linux" on page 748.

Migrating IBM MQ library loading from an earlier version of the product to the latest version

No change in the way IBM MQ libraries are loaded is normally required if you upgrade from an earlier version of the product to the latest version. You must have followed the instructions on building IBM MQ applications in the earlier version, and you must replace the earlier version with the latest version of the product. If you choose to take advantage of multi-installation in the latest version of the product, based on the side-by-side or multi-stage migration scenarios, you must modify the environment for the operating system to resolve IBM MQ dependencies for an application. Typically, you can modify the runtime environment, rather than relink the application.

About this task

This information has moved. See "Migrating IBM MQ library loading to a later version on Windows" on page 709 and "Migrating IBM MQ library loading to a later version on UNIX and Linux" on page 739.

Migrating IBM MQ on Windows

Windows

IBM MQ migration tasks associated with Windows platforms are grouped in this section.

Procedure

- For information about creating a migration plan, see "Planning to migrate to a later version on Windows" on page 685.
- For information about migrating a queue manager from an earlier version to the latest version, see "Migrating a queue manager to a later version on Windows" on page 691.
- For information about reverting a queue manager to an earlier version, see "Reverting a queue manager to an earlier version on Windows" on page 705.
- For information about migrating an IBM MQ MQI client to the latest version, see "Migrating an IBM MQ MQI client to a later version on Windows" on page 707.
- For information about converting a single instance queue manager to a multi-instance queue manager, see "Migrating from a single instance to a multi-instance queue manager on Windows" on page 712.

- For information about reverting a multi-instance queue manager to a single instance queue manager, see “Reverting to a single-instance queue manager on Windows” on page 715.
- For information about migrating IBM MQ library loading to the latest version, see “Migrating IBM MQ library loading to a later version on Windows” on page 709.
- For information about migrating MQ Telemetry to the latest version, see “Migrating MQ Telemetry on Windows” on page 716.
- For information about migrating an MSCS configuration to the latest version, see “Migrating an MSCS configuration on Windows” on page 717.
- V 9.0.4 For information about Migrating logs to an Advanced Format disk, see “Migrating logs to an Advanced Format disk on Windows” on page 719.

Related concepts:

“Migration concepts and methods” on page 645
An overview of the various concepts and methods for migrating from one release of the product to another.

Related tasks:

UNIX Linux “Migrating IBM MQ on UNIX and Linux” on page 720
Migration tasks associated with UNIX and Linux platforms are grouped in this section.

IBM i “Migrating IBM MQ on IBM i” on page 749
IBM MQ migration tasks associated with IBM i are grouped in this section.

z/OS “Migrating IBM MQ on z/OS” on page 770
Migration tasks associated with z/OS are grouped in this section.

Related reference:

“Changes that affect migration” on page 626
Changes to the product might affect the migration of a queue manager from an earlier release to the current release of IBM MQ, or affect existing applications or configurations. Review these changes before upgrading queue managers to the latest product version and decide whether you must plan to make changes to existing applications, scripts, and procedures before starting to migrate your systems.

Planning to migrate to a later version on Windows

Windows

Before migrating from one version of IBM MQ to another on Windows, read this topic, then create your own migration plan based on the outline in this topic.

If there are concepts about migration you do not understand, read “Migration concepts and methods” on page 645, and the subsections you need, first.

About this task

Use the following steps as a guide to creating a migration plan.

Procedure

1. Review the IBM MQ system requirements for the later version of the product. See System Requirements for IBM MQ.
2. Decide whether to run the earlier version and the later version of the product on the same server, and also which migration method you want to use. See “Migration methods on IBM MQ for Multiplatforms” on page 653.

   Important: Note that if you require a multi-version installation, and you are using a version of the product prior to IBM WebSphere MQ Version 7.0.1, Fix Pack 6, you must install Version 7.0.1, Fix Pack 6 first before installing the later version.

3. Review all the changes in IBM MQ that affect you. For more information, see “Changes that affect migration” on page 626.
4. Review performance changes. See IBM MQ Family - Performance Reports.
5. Review the readme file for the product that you are working with. See IBM MQ, WebSphere MQ, and MQSeries product readmes.
6. Plan the sequence and timing of queue manager upgrades.
   - If the queue manager is part of a queue manager cluster, you must migrate the queue managers that are full repositories first.
   - If the queue manager is part of a high availability cluster, plan the migration to minimize downtime and maximize availability; see “Migrating a queue manager in a high-availability configuration” on page 814.
7. Plan to migrate your queue manager to the later version. See “Migrating a queue manager to a later version on Windows” on page 691.

   Backing up queue manager data is part of the queue manager migration task. An alternative approach to backing up queue manager data, is to install and configure a new server. Test the later version with a new queue manager on the new server. When you are ready to go into production on the later version, copy the queue manager configuration and data, to the new server.
8. Plan to update any manual or automated procedures you have written with changes to messages and codes.
9. Decide on what regression tests to perform before putting the queue manager into production on the later version. Include the procedures and applications you identified in steps 6 and 7 in your regression tests.
10. Plan to upgrade your IBM MQ MQI client installations to the later version.
11. Plan to upgrade your client and server applications to use new functions in the later version.

Migration considerations for Version 8.0 or later on Windows: Windows

From Version 8.0, a number of changes were made for IBM MQ for Windows. You must understand these changes before planning any migration tasks for Version 8.0 or later on Windows.

Installing a single copy of the product

If you have an existing previous version of the product on your system, and want to upgrade to the latest version, you have various options. You can either:
- Uninstall the previous version and then install the latest version,
- Install the new copy alongside the currently installed one and uninstall the original at a later time. See “Installing the product alongside an existing version,” or
- Perform a migration installation, electing to replace the currently installed version when prompted.

After you have installed the product, start each queue manager and its data migration takes place. This includes the migration of queue managers from 32-bit to 64-bit.

Installing the product alongside an existing version

If you want to install another version of the product alongside your existing product you can do so. See “Multiple IBM MQ installations” on page 660 and “Migrating on Windows: side-by-side” on page 699 for further information.

When you install the new version of the product, run the setmqm command to associate the queue managers with the new installation.

Start each queue manager in turn and its data migration takes place.

Upgrading one of a pair of (or more) installations

If you already have, for example, an IBM MQ Version 8.0 installation and an IBM MQ Version 9.0 installation on a machine, upgrading the Version 8.0 installation to Version 9.0 requires the following additional step.

When you start the Version 9.0 installer, you are asked whether you want to **Install a new instance** or **Maintain or upgrade an existing instance**.

However, only the other Version 9.0 installation, or installations, are displayed; not the Version 8.0 installation in the selection box. At this point, select **Install a new instance**.

After the splash screen has been displayed, a second panel appears, which lists any older installations that you can upgrade to Version 9.0 using the Version 9.0 installer.

On this panel, select **Upgrade 8.0.0.n Installation 'Installation m'**, and then click **Next**.

Change of digital signature algorithm

The IBM MQ programs and installation image are digitally signed on Windows to confirm that they are genuine and unmodified.

In older releases before IBM MQ Version 8.0, the product was signed using the SHA-1 with RSA algorithm.

From Version 8.0, the SHA-256 with RSA algorithm is used. Some older versions of Windows do not support the new digital signature algorithm, but those versions are not supported by IBM MQ Version 8.0 or later.

See Hardware and software requirements on Windows systems, and ensure that you install IBM MQ Version 8.0 or later on a supported version of Windows.

Existing applications

All applications that were built with previous versions of the product continue to work in Version 8.0 or later with a 64 bit queue manager.

All applications using the C++ object interface need to be rebuilt; applications using the C interface are not affected.

Exits

Queue manager exits on Windows 64-bit operating systems must be compiled as 64-bit exits. Any 32-bit queue manager exits must be recompiled before they can be used with a 64-bit queue manager. If you try to use a 32-bit exit with a 64-bit queue manager on IBM MQ Version 8.0 or later, an AMQ9535 "invalid exit" error message is issued.

Clients

32-bit client applications can connect transparently to queue managers from all supported versions of the product. This includes 64-bit IBM MQ Version 8.0 or later.

Samples

From IBM MQ Version 8.0, the samples for the C and C++ languages are compiled as 64-bit.

Related concepts:

"changes from IBM MQ Version 8.0 on Windows" on page 632
From IBM MQ Version 8.0, changes have been made to the default installation location for the server and 64-bit client installations. There has also been a change in the default data path for all installations. The compiler that is used from IBM MQ Version 8.0 on Windows has been changed.

Related information:

Directory structure on Windows systems

Hardware and software requirements on Windows systems

Program and data directory locations on Windows: Windows

The installation location for IBM MQ program binary and data files on Windows depends on the IBM MQ version you are installing and whether this is the first time IBM MQ is being installed.

First-time installations

When you install IBM MQ for the first time, you can accept default installation locations. You can also select the custom installation option by choosing the location for the IBM MQ binary files and the location for the IBM MQ data and logs.

Before Version 8.0, if the default option was chosen, both the IBM MQ program binary and the data files were installed in the same directory. From Version 8.0, default locations are changed.

Table 83. Default program and data directory locations on different versions of IBM MQ on Windows

IBM MQ Version	IBM MQ program binary files installation location	IBM MQ data files location
IBM WebSphere MQ Version 7.0.1, Version 7.1, and Version 7.5	Program and data files are in one location: `C:\Program Files (x86)\IBM\WebSphere MQ`	
IBM MQ Version 8.0	`C:\Program Files\IBM\WebSphere MQ`	`C:\ProgramData\IBM\MQ`
IBM MQ Version 9.0	`C:\Program Files\IBM\MQ`	`C:\ProgramData\IBM\MQ`

Subsequent installations and reinstallations

After the data directory is specified, during the installation process of any installation, it cannot be changed for subsequent installations. IBM MQ is only installed as a 64-bit version when it is installed on a 64-bit operating system.

For Version 9.0, the default data directory is `C:\ProgramData\IBM\MQ`, unless a version of the product was previously installed, in which case the new installation continues to use the existing data directory.

Existing IBM WebSphere MQ Version 7.0.1 installation

By default, the Version 7.0.1 data and program binary files are installed into `C:\Program Files (x86)\IBM\WebSphere MQ`.

Three upgrade paths are possible:

- Uninstall Version 7.0.1 first and then install Version 9.0.
- Upgrade Version 7.0.1 at the beginning of the Version 9.0 installation process, without first uninstalling the earlier version.
- Install Version 9.0 alongside Version 7.0.1, Fix Pack 6 or later and then uninstall Version 7.0.1.

Installing Version 9.0 after uninstalling Version 7.0.1
If you install IBM MQ Version 9.0 after uninstalling Version 7.0.1, the installation program looks

in the `C:\ProgramData\IBM\MQ` directory for any existing data, which will not exist. The Version 9.0 program files are installed into `C:\Program Files\IBM\MQ` and the program data by default uses `C:\ProgramData\IBM\MQ`.

If data exists from a previous installation, you must choose the custom installation option and manually set the data path to the location that Version 7.0.1 was previously using, as the data path for the new installation, in order to pick up any existing queue managers.

Upgrading Version 7.0.1 at the beginning of the Version 9.0 installation process

If you install Version 9.0 without uninstalling Version 7.0.1 and choose to upgrade the Version 7.0.1 installation, the new program binary files replace the Version 7.0.1 binary files, hence by default the location of the binary files is `C:\Program Files (x86)\IBM\WebSphere MQ`.

The existing data path is kept so, by default, the data path is also `C:\Program Files (x86)\IBM\WebSphere MQ`.

Installing Version 9.0 to coexist with Version 7.0.1

If you install Version 9.0 without uninstalling Version 7.0.1 and choose to install alongside the Version 7.0.1 installation, the new program binary files are put in the new default location `C:\Program Files\IBM\MQ`.

The existing data path is kept so, by default, the data path is `C:\Program Files (x86)\IBM\WebSphere MQ`.

Existing IBM WebSphere MQ Version 7.5 or Version 7.1 installation

Three upgrade paths are possible:

- Uninstall Version 7.1 or Version 7.5 first and then install the latest version.
- Upgrade Version 7.1 or Version 7.5 at the beginning of the Version 9.0 installation process, without first uninstalling the earlier version.
- Install Version 9.0 alongside Version 7.1 or Version 7.5 and then uninstall the earlier version.

When IBM WebSphere MQ Version 7.5 or Version 7.1 is installed, both the program binary files and data are installed by default into `C:\Program Files (x86)\IBM\WebSphere MQ`.

When you uninstall IBM WebSphere MQ Version 7.5 or Version 7.1, information about the location of the data directory is left in the registry.

Installing Version 9.0 after uninstalling IBM WebSphere MQ Version 7.5 or Version 7.1

After uninstalling Version 7.5 or Version 7.1, Version 9.0 is installed using the same installation name but using the Version 9.0 default program binary files location of `C:\Program Files\IBM\MQ`. That is, the program files move from the Windows 32-bit program location to the Windows 64-bit program location.

Optional: You can use the custom installation option to modify the installation path, including modifying it back to `C:\Program Files (x86)\IBM\WebSphere MQ`.

Upgrading IBM WebSphere MQ Version 7.5 or Version 7.1 at the beginning of the Version 9.0 installation process

If you install Version 9.0 without uninstalling Version 7.1 or Version 7.5 and choose to upgrade the Version 7.1 or Version 7.5 installation, the new program binary files replace the Version 7.1 or Version 7.5 binary files so, by default, the new binary files are in `C:\Program Files (x86)\IBM\WebSphere MQ`. The existing data path is kept so, by default, the data path is also `C:\Program Files (x86)\IBM\WebSphere MQ`.

Installing Version 9.0 to coexist with IBM WebSphere MQ Version 7.5 or Version 7.1

If you install Version 9.0 alongside Version 7.1 or Version 7.5, a unique path is chosen, which by default is `C:\Program Files\IBM\MQ`. The existing data path is kept so, by default, the data path is `C:\Program Files (x86)\IBM\WebSphere MQ`.

Existing IBM MQ Version 8.0 installation

Three upgrade paths are possible:

- Uninstall IBM MQ Version 8.0 first and then install Version 9.0.
- Upgrade IBM MQ Version 8.0 at the beginning of the Version 9.0 installation process, without first uninstalling the earlier version.
- Install Version 9.0 alongside IBM MQ Version 8.0 and then uninstall IBM MQ Version 8.0.

When IBM MQ Version 8.0 is installed, the product binary files are put by default into `C:\Program Files\IBM\WebSphere MQ` and the product data and logs are put by default into `C:\ProgramData\IBM\MQ`.

When you uninstall IBM MQ Version 8.0, information about the location of the data directory is left in the registry. After uninstalling IBM MQ Version 8.0 and before installing Version 9.0, you can run the `ResetMQ.cmd` script to tidy up files and data left behind by the uninstallation process.

Important: You should use this script with caution. `ResetMQ.cmd` can remove the existing queue manager configuration. For more information, read Clearing IBM MQ installation settings.

Installing Version 9.0 after uninstalling IBM MQ Version 8.0

After uninstalling IBM MQ Version 8.0, the Version 9.0 is installed using the same installation name but using the Version 9.0 default program binary files location `C:\Program Files\IBM\MQ`. That is, the program files change location after the upgrade.

Optional: You can use the custom installation option to modify the installation path, including modifying it back to `C:\Program Files (x86)\IBM\WebSphere MQ`.

The default data path is `C:\ProgramData\IBM\MQ`.

Upgrading IBM MQ Version 8.0 at the beginning of the Version 9.0 installation process

If you install Version 9.0 without uninstalling IBM MQ Version 8.0 and choose to upgrade the IBM MQ Version 8.0 installation, the new program binary files replace the IBM MQ Version 8.0 binary files so, by default, the new binary files are in `C:\Program Files (x86)\IBM\WebSphere MQ`. The existing data path is kept so, by default, the data path is `C:\ProgramData\IBM\MQ`.

Installing Version 9.0 to coexist with IBM MQ Version 8.0

If you install Version 9.0 alongside IBM MQ Version 8.0, a unique path is chosen, which by default is `C:\Program Files\IBM\MQ`. The existing data path is kept so, by default, the data path is `C:\ProgramData\IBM\MQ`.

Related concepts:

"Migration concepts and methods" on page 645
An overview of the various concepts and methods for migrating from one release of the product to another.

Related information:

Clearing IBM MQ installation settings

Hardware and software requirements on Windows systems

Migrating a queue manager on Windows

Windows

The procedures for migrating a queue manager to a later version of the product, and for restoring a queue manager to an earlier version of the product are detailed in this section.

Related tasks:

UNIX Linux "Migrating a queue manager on UNIX and Linux" on page 721
The procedures for migrating a queue manager to a later version of the product, and for restoring a queue manager to an earlier version of the product are detailed in this section.

IBM i "Migrating a queue manager to a later version on IBM i" on page 751
Follow these instructions to migrate a queue manager from an earlier release to a later release.

z/OS "Migrating IBM MQ on z/OS" on page 770
Migration tasks associated with z/OS are grouped in this section.

Migrating a queue manager to a later version on Windows: Windows

On Windows platforms, follow these instructions to migrate a queue manager from an earlier version to a later version of IBM MQ.

Before you begin

If you have installed early support program code on the server, you must delete all the queue managers created with the installation. Uninstall the code before proceeding with installing the production level code.

1. Create a migration plan; see "Planning to migrate to a later version on Windows" on page 685.
2. Review the IBM MQ system requirements for the latest version, including information about the versions of Windows that IBM MQ supports. See System Requirements for IBM MQ.
3. Back up your system before you install a later version of IBM MQ over an earlier version. Once you have started a queue manager you cannot revert to the previous version. If you must restore the system, you cannot recover any work, such as changes to messages and objects, performed by the later version of IBM MQ. For more information about backing up your system, see Backing up and restoring IBM MQ queue manager data.
4. Review any other installed SupportPacs for their applicability to the later version.
5. If you are running on a server with multiple IBM MQ installations, you must identify the installation. Make sure that the commands you enter run against the correct installation; see setmqenv.

About this task

To run a command, the operating system must find the command in an IBM MQ installation. For some commands, you must run the command from the installation that is associated with the correct queue manager. IBM MQ does not switch commands to the correct installation. For other commands, such as **setmqinst**, you can run the command from any installation that has the later version of the product installed.

If an earlier version of the product is installed, the command that is run is the command for that version, unless the search path is overridden by a local setting. You can override the search path by running **setmqenv**. If Version 7.0.1 is not installed, you must set the correct path to run a command. If you have set a primary installation, the command that is run is the copy in the primary installation, unless you override the selection with a local search path.

Procedure

1. Log in as a user in `group mqm`.
2. Stop all applications using the IBM MQ installation.

   If you use the Managed File Transfer (MFT) component, ensure that any MFT agents have finished all of the file transfers that they were engaged in. There should be no incomplete transfers associated with the agents, and their SYSTEM.FTE.STATE queues should contain no messages.
3. End all the activity of queue managers associated with the IBM MQ installation.
   a. Run the **dspmq** command to list the state of all the queue managers on the system.

      Run either of the following commands from the installation that you are updating:

      ```
      dspmq -o installation -o status
      dspmq -a
      ```

      dspmq -o installation -o status displays the installation name and status of queue managers associated with all installations of IBM MQ.

      dspmq -a displays the status of active queue managers associated with the installation from which the command is run.
   b. Use the MQSC command **DISPLAY LSSTATUS** to list the status of listeners associated with a queue manager, as shown in the following example:

      ```
      echo "DISPLAY LSSTATUS(*) STATUS" | runmqsc QmgrName
      ```
   c. Run the **endmqm** command to stop each running queue manager associated with this installation.

      ```
                  ┌-c┐
      ►►──endmqm──┼-w┼──QmgrName──────────────────────────────────────►◄
                  ├-i┤
                  └-p┘
      ```

      The **endmqm** command informs an application that the queue manager it is connected to is stopping; see Stopping a queue manager.

      For the maintenance to proceed, applications must respond to an **endmqm** command by disconnecting from the queue manager and releasing any IBM MQ libraries they have loaded. If they do not, you must find another way to force applications to release IBM MQ resources, such as by stopping the applications.

      You must also stop applications that are using the client libraries that are part of the installation. Client applications might be connected to a different queue manager, running a different installation of IBM MQ. The application is not informed about queue managers in the current installation being shut down.

      Any applications that continue to have IBM MQ shared libraries from the installation loaded prevent you applying IBM MQ maintenance. An application might disconnect from a queue manager, or be forcibly disconnected, but keep an IBM MQ shared library loaded.

      Note: "Applying maintenance level updates to multi-instance queue managers on Windows" on page 577 describes how to apply maintenance to a multi-instance queue manager. A multi-instance queue manager can continue to run on one server, while maintenance is applied to another server.
   d. Stop any listeners associated with the queue managers, using the command:

      ```
      endmqlsr -m QMgrName
      ```

4. Back up the queue manager. Take copies of all the queue manager's data and log file directories, including all subdirectories, and also the `qm.ini` file and the registry entries. For more information, see Backing up and restoring IBM MQ queue manager data.
5. Stop the IBM WebSphere MQ or IBM MQ Service and exit the Service icon application.
6. Optional: If you are doing a single stage migration and you are migrating from IBM WebSphere MQ Version 7.0.1, Fix Pack 6 or later, optionally uninstall the current version of the product. Note, that you carry out this step only if you are doing a single stage migration; see “Migrating on Windows: single stage” on page 694.
7. Install the later version of IBM MQ. On Windows, you can do this either by using the Installation Launchpad or by using the **`msiexec`** command. For more information, see:
   - Modifying the installation using IBM MQ Installation Launchpad
   - Silently modifying an IBM MQ server installation using **`msiexec`**
8. Reenter domain, user ID, and password information

   When the installation of the latest version completes, the Prepare IBM MQ Wizard starts automatically.

   Where UAC is enabled: If you rerun the Prepare IBM MQ Wizard, ensure that the wizard is run with Administrator privilege, otherwise the wizard might fail.
9. Start the queue manager.

   ```
   strmqm QmgrName
   ```

   When you first start a queue manager after migration:
   - Any new attributes for existing objects are set to their default values.
   - Any new default objects are created.
   - Queue manager data is migrated.

   Important: Do not use the `-c` option to start the queue manager, unless you explicitly want to reset or re-create the default system objects.

   You must start IBM MQ before you start any listeners.

What to do next

Complete the tasks in your migration plan, such as verifying the new code level and deploying new functions such as automatically restarting client connections.

If you are using publish/subscribe, you must migrate the publish/subscribe broker.

If the queue manager is a member of a queue manager cluster, migrate the other members of the cluster.

Important: You must migrate the publish/subscribe broker state before you migrate your IBM MQ system to IBM MQ Version 8.0, or later, as broker publish/subscribe migration is not supported in IBM MQ Version 8.0, or later.

➡ Fix Central

➡ IBM Passport Advantage

"Migrating a queue manager in a high-availability configuration" on page 814
High-availability configurations of queue managers can increase the availability of IBM MQ applications. If a queue manager, or server fails, it is restarted automatically on another server. You can arrange for IBM MQ MQI client applications to automatically reconnect to the queue manager. Server applications can be configured to start when the queue manager starts.

"Migrating a queue manager cluster" on page 808
You can migrate queue managers in a cluster all at once, or one at a time, which is called a staged migration. Migrate full repository queue managers in a cluster before partial repository queue managers. You must consider what the effect is of migrating some queue managers in a cluster, before all the queue managers are migrated.

"Restoration of a queue manager to an earlier version on all platforms" on page 650
You can remove an upgrade before you have started a queue manager. However, if you remove the upgrade after a queue manager has been started, the queue manager will not work.

➡ IBM MQ - SupportPacs by Product

"Maintaining and migrating IBM MQ" on page 565
Maintenance, upgrade, and migration have three distinct meanings for IBM MQ. The definitions are described here. The following sections describe the various concepts associated with migration, followed by the various tasks needed; these tasks are platform-specific where needed.

"Migrating IBM MQ" on page 625
Migration is the conversion of programs and data to work with a new code level of IBM MQ. Some types of migration are required, and some are optional. Queue manager migration is never required after applying a maintenance level update, that does not change the command level. Some types of migration are automatic, and some are manual. Queue manager migration is typically automatic and required after releases and manual and optional after a maintenance level upgrade that introduces a new function. Application migration is typically manual and optional.

"Applying upgrades and fixes to IBM MQ" on page 618
The term upgrade applies to changing the version `V`, release `R`, or modification `M` of a product. The term fix applies to a change in the `F` digit.

Migrating on Windows: single stage: Windows

Single-stage migration is the term used to describe replacing the only installation of IBM MQ on a server, with a later version of the product. Single stage migration is also known as *upgrading in place* or *in place upgrade*. Until Version 7.0.1, Fix Pack 6, single-stage was the only migration scenario. Single-stage migration preserves existing scripts and procedures for running IBM MQ the most. With other migration scenarios you might change some scripts and procedures, but you can reduce the effect queue manager migration has on users.

Before you begin

These topics guide you in deciding what other tasks you must perform to migrate queue managers and applications to the later version. For the precise sequence of commands to upgrade a queue manager to the later version, do the migration task for the platform you are interested in. All the tasks are listed by platform in the links at the end of this topic. As part of the queue manager migration task, back up your existing queue manager data. Even on a multi-installation server, queue managers cannot be restored to a previous command level after migration.

Attention: V 9.0.0 From IBM MQ Version 9.0, the `ccsid_part2.tbl` file replaces the existing `ccsid.tbl` file, used in previous versions of the product, to supply additional CCSID information.

The `ccsid_part2.tbl` file takes precedence over the `ccsid.tbl` file and:
- Allows you to add or modify CCSID entries
- Specify default data conversion
- Specify data for different command levels

The `ccsid_part2.tbl` is applicable to the following platforms only:
- Linux - all versions
- Solaris
- Windows

If you have added any of your own CCSID information into your existing `ccsid.tbl` file, you should copy this information into the new `ccsid_part2.tbl` file, if you want to take advantage of the new formats in your customizations

You should copy the required information, rather than move the information, so that your existing version of IBM MQ continues to work.

About this task

In the single-stage migration scenario, the installation of the later version of the product replaces an earlier version in the same installation location. It is the same migration process that you might previously have used to upgrade the product prior to IBM WebSphere MQ Version 7.0.1, Fix Pack 6. From Version 7.0.1, Fix Pack 6, this method of migration is termed *single-stage* migration, in contrast to *side-by-side* and *multi-stage* migration.

The advantage of single-stage migration is that it changes the configuration of a queue manager on the earlier version as little as possible. Existing applications switch from loading the libraries from the earlier version, to loading the libraries of the later version, automatically. Queue managers are automatically associated with the installation on the later version. Administrative scripts and procedures are affected as little as possible by setting the installation to be the primary installation. If you set the installation of the later version to be the primary installation, commands such as **`strmqm`** work without providing an explicit path to the command.

When you upgrade the earlier version to the later version, all the objects that you previously created are maintained. The components that were previously installed are preselected in the feature options when you install the new level. If you leave these components selected, you can keep them or reinstall them. If you clear any of these components, the installation process uninstalls them. By default, a typical migration installs only the same features that were installed in the previous version installation.

For example, if IBM MQ Explorer was not installed in an earlier installation, it is not stored in a later installation. If you want IBM MQ Explorer, select a custom installation, and select the IBM MQ Explorer feature on the Features panel. If you do not want IBM MQ Explorer, uninstall the IBM MQ Explorer feature by selecting a custom installation. Then clear the IBM MQ Explorer feature on the Features panel. For more information about how to uninstall features, see Modifying the installation using IBM MQ Installation Launchpad.

You can also migrate a queue manager to a later version of the product on a system where an earlier version has been uninstalled. In this case, the queue manager data must have been retained, or restored from a backup.

Procedure
1. Log in as a user in `group mqm`.
2. Stop all applications using the IBM MQ installation.

If you use the Managed File Transfer (MFT) component, ensure that any MFT agents have finished all of the file transfers that they were engaged in. There should be no incomplete transfers associated with the agents, and their SYSTEM.FTE.STATE queues should contain no messages.

3. End all the activity of queue managers associated with the IBM MQ installation.
   a. Run the **dspmq** command to list the state of all the queue managers on the system.

      Run either of the following commands from the installation that you are updating:

      ```
      dspmq -o installation -o status
      dspmq -a
      ```

      dspmq -o installation -o status displays the installation name and status of queue managers associated with all installations of IBM MQ.

      dspmq -a displays the status of active queue managers associated with the installation from which the command is run.
   b. Use the MQSC command **DISPLAY LSSTATUS** to list the status of listeners associated with a queue manager, as shown in the following example:

      ```
      echo "DISPLAY LSSTATUS(*) STATUS" | runmqsc QmgrName
      ```
   c. Run the **endmqm** command to stop each running queue manager associated with this installation.

      ```
                    ┌─-c─┐
      ►►──endmqm────┼─-w─┼──QmgrName──────────────────────────────────────────────►◄
                    ├─-i─┤
                    └─-p─┘
      ```

      The **endmqm** command informs an application that the queue manager it is connected to is stopping; see Stopping a queue manager.

      For the maintenance to proceed, applications must respond to an **endmqm** command by disconnecting from the queue manager and releasing any IBM MQ libraries they have loaded. If they do not, you must find another way to force applications to release IBM MQ resources, such as by stopping the applications.

      You must also stop applications that are using the client libraries that are part of the installation. Client applications might be connected to a different queue manager, running a different installation of IBM MQ. The application is not informed about queue managers in the current installation being shut down.

      Any applications that continue to have IBM MQ shared libraries from the installation loaded prevent you applying IBM MQ maintenance. An application might disconnect from a queue manager, or be forcibly disconnected, but keep an IBM MQ shared library loaded.

      Note: "Applying maintenance level updates to multi-instance queue managers on Windows" on page 577 describes how to apply maintenance to a multi-instance queue manager. A multi-instance queue manager can continue to run on one server, while maintenance is applied to another server.
   d. Stop any listeners associated with the queue managers, using the command:

      ```
      endmqlsr -m QMgrName
      ```
4. Back up the queue manager. Take copies of all the queue manager's data and log file directories, including all subdirectories, and also the `qm.ini` file and the registry entries. For more information, see Backing up and restoring IBM MQ queue manager data.
5. Stop the IBM WebSphere MQ or IBM MQ Service and exit the Service icon application.
6. Optional: If you are migrating from IBM WebSphere MQ Version 7.0.1, Fix Pack 6 or later, optionally uninstall the current version of the product.
7. Upgrade the earlier version of the product to the later version in the same installation directory. A reason for installing into the same location is to simplify application migration. If you change the

installation location, you might remove IBM MQ libraries from an application search path. To migrate an application search path you must modify the application environment, or more rarely, the application itself.

a. Decide on an installation naming convention. Give the installation a name of your choosing, or accept the default installation name. For the first installation, the default name is *Installation1*. For the second installation, the name is *Installation2*, and so on.
b. Upgrade the earlier version of the product to the later version in place, or uninstall the earlier version, without deleting any queue managers, and install the later version in the same default location. On Windows, you can do this either by using the Installation Launchpad or by using the **msiexec** command. For more information, see:
   - Modifying the installation using IBM MQ Installation Launchpad
   - Silently modifying an IBM MQ server installation using **msiexec**

   On Windows, uninstalling the previous version of the product before you install the later version is optional.

8. Reenter domain, user ID, and password information

   When the installation of the latest version completes, the Prepare IBM MQ Wizard starts automatically.

   Where UAC is enabled: If you rerun the Prepare IBM MQ Wizard, ensure that the wizard is run with Administrator privilege, otherwise the wizard might fail.
9. Optional: Make the later version of the installation the primary installation.
   a. Run the **setmqinst** command

   ```
   "Inst_1_INSTALLATION_PATH\bin\setmqinst" -i -n Inst_1
   ```

   Make the installation primary to avoid specifying a search path to run IBM MQ commands
10. Start the queue managers and applications.
    a. Run the **setmqm** command to associate the queue managers with `Inst_1`.

    ```
    setmqm -m QM1 -n Inst_1
    setmqm -m QM2 -n Inst_1
    ```

    Notes:
    - The **setmqm** step is optional only in the case where migration is from IBM WebSphere MQ Version 7.0.1 to a later release. In this case, the **strmqm** command automatically associates the queue manager with its own installation.
    - If you are migrating between any other releases of the product, you must use **setmqm** to associate the queue managers with the new installation manually.

    If you have multiple installations, note that queue managers that were configured to start automatically, and remain after uninstalling IBM WebSphere MQ Version 7.0.1, automatically start under any other existing Version 7.1 (or later) installation when either the machine reboots, or the Service for that installation is restarted. In order to avoid this, ensure that all queue managers have been moved to the required installation before uninstalling IBM WebSphere MQ Version 7.0.1.

    b. Run the **strmqm** command to start the queue managers and migrate them to the later version of the product.

    ```
    strmqm QM1
    strmqm QM2
    ```

    You must start IBM MQ before you start any listeners.

    When you first start a queue manager after migration:
    - Any new attributes for existing objects are set to their default values.
    - Any new default objects are created.
    - Queue manager data is migrated.

At this point, when the queue manager data is migrated, you cannot revert to a previous release.

Important: Do not use the `-c` option to start the queue manager, unless you explicitly want to reset or re-create the default system objects.

- When an application connects to a queue manager, the operating system searches its load path to load the IBM MQ library [4] . A Version 7.1, or later, library contains code that checks that the queue manager is associated with an installation. If a queue manager is associated with a different installation, IBM MQ loads the correct IBM MQ library for the installation the queue manager is associated with.

What to do next

You cannot reinstall an earlier version of the product on a system that has the latest, or any other, version of IBM MQ installed.

Related concepts:

"Queue manager coexistence" on page 664
Queue managers, with different names, can coexist on any server as long as they use the same IBM MQ installation. On z/OS, UNIX, Linux, and Windows, different queue managers can coexist on the same server and be associated with different installations.

"Multi-installation queue manager coexistence on UNIX, Linux, and Windows" on page 668
You can install multiple copies of IBM MQ for UNIX, Linux, and Windows on the same server. The installations must be at Version 7.1 or later, with one exception. One Version 7.0.1 installation, at fix pack level 6, or later, can coexist with multiple Version 7.1, or later installations.

Related tasks:

"Planning to migrate to a later version on Windows" on page 685
Before migrating from one version of IBM MQ to another on Windows, read this topic, then create your own migration plan based on the outline in this topic.

"Migrating a queue manager to a later version on UNIX and Linux" on page 722
On UNIX and Linux, follow these instructions to migrate a queue manager from an earlier version to a later version of IBM MQ.

"Migrating a queue manager to a later version on Windows" on page 691
On Windows platforms, follow these instructions to migrate a queue manager from an earlier version to a later version of IBM MQ.

"Migrating IBM MQ library loading from an earlier version of the product to the latest version" on page 684
No change in the way IBM MQ libraries are loaded is normally required if you upgrade from an earlier version of the product to the latest version. You must have followed the instructions on building IBM MQ applications in the earlier version, and you must replace the earlier version with the latest version of the product. If you choose to take advantage of multi-installation in the latest version of the product, based on the side-by-side or multi-stage migration scenarios, you must modify the environment for the operating system to resolve IBM MQ dependencies for an application. Typically, you can modify the runtime environment, rather than relink the application.

Related information:

Installing IBM MQ server on Windows

Associating a queue manager with an installation

Changing the primary installation

Choosing an installation name

setmqenv

4. On Windows, the IBM MQ library is a DLL. A DLL is sometimes called a load library or a shared library. The entry points to a DLL are defined in a link library, with the file extension `.lib32` or `.lib`. The `.lib` library is linked at build-time and the DLL loaded at runtime.

setmqinst

setmqm

Migrating on Windows: side-by-side: Windows

Side-by-side migration is the term used to describe installing a later version of IBM MQ alongside an earlier version on the same server. Queue managers remain running during the installation and verification of the later version of IBM MQ. They remain associated with the older version of IBM MQ. When you decide to migrate queue managers to the later version of IBM MQ, you stop all queue managers, uninstall the earlier version, and migrate them all to the new version of IBM MQ.

Before you begin

If you are using IBM WebSphere MQ Version 7.0.1, you must ensure that you are running IBM WebSphere MQ Version 7.0.1, Fix Pack 6 or later before installing the later version of the product on the same server. For more information about Version 7.0.1 fix packs, see Recommended Fixes for IBM MQ.

Attention: V 9.0.0 From IBM MQ Version 9.0, the `ccsid_part2.tbl` file replaces the existing `ccsid.tbl` file, used in previous versions of the product, to supply additional CCSID information.

The `ccsid_part2.tbl` file takes precedence over the `ccsid.tbl` file and:
- Allows you to add or modify CCSID entries
- Specify default data conversion
- Specify data for different command levels

The `ccsid_part2.tbl` is applicable to the following platforms only:
- Linux Linux - all versions
- Solaris Solaris
- Windows Windows

If you have added any of your own CCSID information into your existing `ccsid.tbl` file, you should copy this information into the new `ccsid_part2.tbl` file, if you want to take advantage of the new formats in your customizations

You should copy the required information, rather than move the information, so that your existing version of IBM MQ continues to work.

About this task

In the side-by-side migration scenario, you install the later version of IBM MQ alongside queue managers that continue to be associated with Version 7.0.1, or later.

When you are ready to migrate the queue managers, and applications, to the later version:
1. Stop all the queue managers.
2. Uninstall the earlier version of the product.
3. Migrate all the queue managers and applications to the later version.

Procedure
1. Install the later version in a different installation directory from the earlier version.
   a. Decide on an installation naming convention. Give the installation a name of your choosing, or accept the default installation name. For the first installation, the default name is *Installation1*. For the second installation, the name is *Installation2*, and so on.

   b. Verify the installation.

      Run the installation verification procedures and your own tests.

2. Uninstall the earlier version of the product.

   When uninstalling the earlier product, you must stop all queue managers and applications that have loaded an IBM MQ library on the server. For this reason, you might choose to postpone uninstalling the earlier version of the product until a convenient maintenance window. When an earlier version of the product is not installed on a server, it is sufficient to stop the queue managers and applications that have loaded libraries from the installation that you are uninstalling or updating. It is not necessary to stop applications and queue managers associated with other installations.

   a. Stop all applications that have loaded IBM MQ libraries on the server.
   b. Stop the queue managers and listeners on the server.
   c. Uninstall the earlier version of the product.
      - Stop all local IBM MQ applications
      - If you are migrating from IBM WebSphere MQ Version 7.0.1, stop all your queue managers and listeners, ensuring that you do not delete the queue managers.

        Note: If you are migrating from a release other than IBM WebSphere MQ Version 7.0.1 you do not need to stop all the queue managers at this point.

3. Make the later version of the installation the primary installation.

   a. Run the **setmqinst** command

      ```
      "Inst_1_INSTALLATION_PATH\bin\setmqinst" -i -n Inst_1
      ```

      Make the installation primary to avoid specifying a search path to run IBM MQ commands

   Use the dspmqinst command to discover the *Installation name*, or use the default value `Installation 1`.

   Doing this means that you do not have to specify a search path on IBM MQ commands.

4. Start the queue managers and applications.

   - When an application connects to a queue manager, the operating system searches its load path to load the IBM MQ library [5] . A Version 7.1, or later, library contains code that checks that the queue manager is associated with an installation. If a queue manager is associated with a different installation, IBM MQ loads the correct IBM MQ library for the installation the queue manager is associated with.

   During this process you continue to use queue manager QM2 while you upgrade queue manager QM1 and you use queue manager QM1 while you upgrade QM2.

   Note that each queue manager needs to be stopped in order to be associated with the new installation.

What to do next

You cannot reinstall an earlier version of the product on a system that has the latest, or any other, version of IBM MQ installed.

5. On Windows, the IBM MQ library is a DLL. A DLL is sometimes called a load library or a shared library. The entry points to a DLL are defined in a link library, with the file extension `.lib32` or `.lib`. The `.lib` library is linked at build-time and the DLL loaded at runtime.

"Planning to migrate to a later version on Windows" on page 685
Before migrating from one version of IBM MQ to another on Windows, read this topic, then create your own migration plan based on the outline in this topic.

Uninstalling IBM MQ on Windows systems

Installing IBM MQ server on Windows

Associating a queue manager with an installation

Changing the primary installation

Choosing an installation name

"Queue manager coexistence" on page 664
Queue managers, with different names, can coexist on any server as long as they use the same IBM MQ installation. On z/OS, UNIX, Linux, and Windows, different queue managers can coexist on the same server and be associated with different installations.

"Migrating IBM MQ library loading from an earlier version of the product to the latest version" on page 684
No change in the way IBM MQ libraries are loaded is normally required if you upgrade from an earlier version of the product to the latest version. You must have followed the instructions on building IBM MQ applications in the earlier version, and you must replace the earlier version with the latest version of the product. If you choose to take advantage of multi-installation in the latest version of the product, based on the side-by-side or multi-stage migration scenarios, you must modify the environment for the operating system to resolve IBM MQ dependencies for an application. Typically, you can modify the runtime environment, rather than relink the application.

"Multi-installation queue manager coexistence on UNIX, Linux, and Windows" on page 668
You can install multiple copies of IBM MQ for UNIX, Linux, and Windows on the same server. The installations must be at Version 7.1 or later, with one exception. One Version 7.0.1 installation, at fix pack level 6, or later, can coexist with multiple Version 7.1, or later installations.

setmqenv

setmqinst

setmqm

Migrating on Windows: multi-stage: Windows

Multi-stage migration is the term used to describe running a later version of IBM MQ alongside an earlier version on the same server. After installing the later version alongside the earlier version, you can create new queue managers to verify the later installation, and develop new applications. At the same time, you can migrate queue managers and their associated applications from the earlier version to the later version. By migrating queue managers and applications one-by-one, you can reduce the peak workload on staff managing the migration.

Before you begin

If you are using IBM WebSphere MQ Version 7.0.1, you must ensure that you are running IBM WebSphere MQ Version 7.0.1, Fix Pack 6 or later before installing a later version of the product on the same server. For more information about Version 7.0.1 fix packs, see Recommended Fixes for IBM MQ.

Attention: V 9.0.0 From IBM MQ Version 9.0, the `ccsid_part2.tbl` file replaces the existing `ccsid.tbl` file, used in previous versions of the product, to supply additional CCSID information.

The `ccsid_part2.tbl` file takes precedence over the `ccsid.tbl` file and:
- Allows you to add or modify CCSID entries
- Specify default data conversion
- Specify data for different command levels

The `ccsid_part2.tbl` is applicable to the following platforms only:

- Linux Linux - all versions
- Solaris Solaris
- Windows Windows

If you have added any of your own CCSID information into your existing `ccsid.tbl` file, you should copy this information into the new `ccsid_part2.tbl` file, if you want to take advantage of the new formats in your customizations

You should copy the required information, rather than move the information, so that your existing version of IBM MQ continues to work.

Note:

- If an application uses COM or ActiveX it can connect to any queue manager as long as there is a primary installation and it is Version 7.1 or later.
- If you are running the IBM MQ.NET monitor in transactional mode, the queue manager it connects to must be the primary installation.

You cannot migrate these applications to the later version until you uninstall the earlier version.

About this task

In the multi-stage migration scenario, you install the later version of the product alongside running queue managers that continue to be associated with the earlier version. You can create queue managers and run new applications using the later version installation. When you are ready to start migrating queue managers and applications from the earlier, you can do so, one-by-one. When migration to the later version is complete, you can uninstall the earlier version, and make the later version installation the primary installation.

With the multi-stage approach, until you uninstall the earlier version , you must configure an environment to run applications that connect to a queue manager to the later version. You must also provide a path to run IBM MQ commands. Both these tasks are accomplished with the **`setmqenv`** command.

Note: When you have uninstalled the earlier version, and set the later version as a primary installation, in most circumstances it is not necessary to run the **`setmqenv`** command to run applications. It is still necessary to run **`setmqenv`** to set the environment for commands that connect to a queue manager associated with an installation that is not primary.

Procedure

1. Install the later version in a different installation directory from the earlier version and verify the installation.
   a. Decide on an installation naming convention. Give the installation a name of your choosing, or accept the default installation name. For the first installation, the default name is *Installation1*. For the second installation, the name is *Installation2*, and so on.
   b. Verify the installation.
      Run the installation verification procedures and your own tests.
   - You might create new queue managers running the later version, and start to develop new applications before migrating applications from the earlier version.
2. Configure the operating system so that applications load the libraries for the later version of the product.
   a. Migrate queue managers one at a time.

The first set of applications to load the libraries for the later version of the product are the applications that connect to the first queue manager you are going to migrate.

It does not matter if those applications also connect to other queue managers on the server. If the applications load the later version libraries, IBM MQ automatically loads the libraries for the earlier version for those applications that connect to that version.

You can either migrate the operating system environment of all applications, or just the applications that connect to the first queue manager you are going to migrate.

b. Migrate IBM MQ MQI client applications

   Some of the applications might be running as IBM MQ MQI client applications on another workstation. When you migrate a queue manager, clients connected to it continue to run without loading a client library for the later version.

   You can migrate these clients later, when you need to do so.

   Important: If any IBM MQ MQI client applications are using the library for the earlier version on the server, you must eventually migrate the clients to use the later version of the product before you uninstall the earlier version.

3. Migrate an application to load the new library for the later version:
   - Run **setmqenv** to modify the local path that is searched for IBM MQ libraries.
   - Relink applications with an additional runtime load path.

   Consult operating system documentation about how to modify the global search path, or include a fixed runtime load path in the application load module.

   To run **setmqenv** using the `-s` option:

   ```
   "Inst_1_INSTALLATION_PATH\bin\setmqenv" -s
   ```

   The `-s` option sets up the environment for the installation that runs the **setmqenv** command.
4. Restart the queue manager and the applications that connect to it.
   a. Set up the local environment to the installation `Inst_1`.

      ```
      "Inst_1_INSTALLATION_PATH\bin\setmqenv" -s
      ```

      The `-s` option sets up the environment for the installation that runs the **setmqenv** command.
   b. Run the **setmqm** command to associate `QM1` with `Inst_1`.

      ```
      setmqm -m QM1 -n Inst_1
      setmqm -m QM2 -n Inst_1
      ```

   c. Run the **strmqm** command to start `QM1` and migrate it to the later version.

      ```
      strmqm QM1
      strmqm QM2
      ```

   d. Restart application 1

      The application loads the later version library and connects to `QM1`, which is associated with the later version of the product.
5. Migrate all queue managers and applications to the later version.

   Repeat steps 2 on page 702 and 4, when required, until all the queue managers and applications are migrated to the later version of the product.
6. Uninstall the earlier version of the product.

   When uninstalling the earlier product, you must stop all queue managers and applications that have loaded an IBM MQ library on the server. For this reason, you might choose to postpone uninstalling the earlier version of the product until a convenient maintenance window. When an earlier version of the product is not installed on a server, it is sufficient to stop the queue managers and applications that have loaded libraries from the installation that you are uninstalling or updating. It is not necessary to stop applications and queue managers associated with other installations.
   a. Stop all applications that have loaded IBM MQ libraries on the server.
   b. Stop the queue managers and listeners on the server.

c. Uninstall the earlier version of the product.
   - Stop all local IBM MQ applications
   - If you are migrating from IBM WebSphere MQ Version 7.0.1, stop all your queue managers and listeners, ensuring that you do not delete the queue managers.

     Note: If you are migrating from a release other than IBM WebSphere MQ Version 7.0.1 you do not need to stop all the queue managers at this point.

7. Make `Inst_1` the primary installation.
   a. Run the **`setmqinst`** command

   ```
   "Inst_1_INSTALLATION_PATH\bin\setmqinst" -i -n Inst_1
   ```

   Note: Use the dspmqinst command to discover the *`Installation name`*, or use the default value `Installation 1`.

   You do not have to set up a search path to run IBM MQ commands from the primary installation.

What to do next

You cannot reinstall an earlier version of the product on a system that has the latest, or any other, version of IBM MQ installed.

Now that you have uninstalled the earlier version of the product, and made the later installation primary, you can review how the application runtime environment is set. It is no longer necessary to run **`setmqenv`** to set up the search path to load libraries for the later version. If you have only one installation of the later version of the product installed, it is not necessary to run **`setmqenv`** to run commands.

Related concepts:

"Queue manager coexistence" on page 664
Queue managers, with different names, can coexist on any server as long as they use the same IBM MQ installation. On z/OS, UNIX, Linux, and Windows, different queue managers can coexist on the same server and be associated with different installations.

"Multi-installation queue manager coexistence on UNIX, Linux, and Windows" on page 668
You can install multiple copies of IBM MQ for UNIX, Linux, and Windows on the same server. The installations must be at Version 7.1 or later, with one exception. One Version 7.0.1 installation, at fix pack level 6, or later, can coexist with multiple Version 7.1, or later installations.

Related tasks:

"Planning to migrate to a later version on Windows" on page 685
Before migrating from one version of IBM MQ to another on Windows, read this topic, then create your own migration plan based on the outline in this topic.

"Migrating IBM MQ library loading from an earlier version of the product to the latest version" on page 684
No change in the way IBM MQ libraries are loaded is normally required if you upgrade from an earlier version of the product to the latest version. You must have followed the instructions on building IBM MQ applications in the earlier version, and you must replace the earlier version with the latest version of the product. If you choose to take advantage of multi-installation in the latest version of the product, based on the side-by-side or multi-stage migration scenarios, you must modify the environment for the operating system to resolve IBM MQ dependencies for an application. Typically, you can modify the runtime environment, rather than relink the application.

Related information:

Installing IBM MQ server on Windows

Associating a queue manager with an installation

Changing the primary installation

Choosing an installation name

setmqenv

setmqinst

setmqm

Reverting a queue manager to an earlier version on Windows:

On Windows platforms, you can revert a queue manager to an earlier version of the product from a later version, if you have made a backup of the system or queue manager. If you have started the queue manager and processed any messages, or changed the configuration, the task cannot give you any guidance on reverting the current state of the queue manager.

Before you begin

1. You must have made a backup of the system or queue manager before you upgraded to the later version. For more information see Backing up and restoring IBM MQ queue manager data
2. If any messages were processed after starting the queue manager, you cannot easily undo the effects of processing the messages. You cannot revert the queue manager to the earlier version of the product in its current state. The task cannot give you any guidance how to deal with subsequent changes that have occurred. For example, messages that were indoubt in a channel, or in a transmission queue on another queue manager, might have been processed. If the queue manager is part of a cluster, then configuration messages and application messages might have been exchanged.
3. If you are running on a server with multiple IBM MQ installations, you must identify the installation. Make sure that the commands you enter run against the correct installation; see setmqenv.

About this task

When you revert to a earlier version of a queue manager, you revert the queue manager to its earlier code level. Queue manager data is reverted to the state it was in when the queue manager was backed up.

Procedure

1. Log in as a user in `group mqm`.
2. Stop all applications using the IBM MQ installation.

   If you use the Managed File Transfer (MFT) component, ensure that any MFT agents have finished all of the file transfers that they were engaged in. There should be no incomplete transfers associated with the agents, and their SYSTEM.FTE.STATE queues should contain no messages.
3. End all the activity of queue managers associated with the IBM MQ installation.
   a. Run the **`dspmq`** command to list the state of all the queue managers on the system.

      Run either of the following commands from the installation that you are updating:
      ```
      dspmq -o installation -o status
      dspmq -a
      ```
      `dspmq -o installation -o status` displays the installation name and status of queue managers associated with all installations of IBM MQ.

      `dspmq -a` displays the status of active queue managers associated with the installation from which the command is run.
   b. Use the MQSC command **`DISPLAY LSSTATUS`** to list the status of listeners associated with a queue manager, as shown in the following example:
      ```
      echo "DISPLAY LSSTATUS(*) STATUS" | runmqsc QmgrName
      ```
   c. Run the **`endmqm`** command to stop each running queue manager associated with this installation.

```
              ┌─-c─┐
►►──endmqm────┼─-w─┼──QmgrName──────────────────────────────────────────────────►◄
              ├─-i─┤
              └─-p─┘
```

The **endmqm** command informs an application that the queue manager it is connected to is stopping; see Stopping a queue manager.

For the maintenance to proceed, applications must respond to an **endmqm** command by disconnecting from the queue manager and releasing any IBM MQ libraries they have loaded. If they do not, you must find another way to force applications to release IBM MQ resources, such as by stopping the applications.

You must also stop applications that are using the client libraries that are part of the installation. Client applications might be connected to a different queue manager, running a different installation of IBM MQ. The application is not informed about queue managers in the current installation being shut down.

Any applications that continue to have IBM MQ shared libraries from the installation loaded prevent you applying IBM MQ maintenance. An application might disconnect from a queue manager, or be forcibly disconnected, but keep an IBM MQ shared library loaded.

Note: The topic, "Applying maintenance level updates to multi-instance queue managers on Windows" on page 577, describes how to apply maintenance to a multi-instance queue manager. A multi-instance queue manager can continue to run on one server, while maintenance is applied to another server.

d. Stop any listeners associated with the queue managers, using the command:

```
endmqlsr -m QMgrName
```

4. Restore the system, or IBM MQ and the queue manager.

   If your backup procedure was to save the queue manager data, you must reinstall IBM MQ:

   a. Uninstall the earlier installation.
   b. Reinstall the product from a manufacturing refresh.
   c. Apply the fix pack and interim fixes that restore IBM MQ to its previous level.
   d. Restore the queue manager data from the backup taken before installing the later version.
5. Restart the earlier version queue manager.

What to do next

You might be reverting to a earlier version on a server with multiple IBM MQ installations. If one of the installations is primary, after reverting the earlier version that installation, by default, becomes the primary installation.

You must review how applications connect to an installation. After reverting to the earlier version, some applications might connect to the wrong installation.

Related information:

Backing up and restoring a queue manager

BFGSS0023E errors and how to avoid them

Migrating an IBM MQ MQI client on Windows

▶ Windows

Before migrating an IBM MQ MQI client, create a migration plan. Stop all IBM MQ activity on the client workstation. Upgrade the IBM MQ MQI client installation. Make any essential configuration and application changes.

Related concepts:

"IBM MQ MQI client migration" on page 650
IBM MQ MQI client migration is the process of converting IBM MQ MQI client configurations, and client and server channels from one version to another. Client migration can take place after upgrading the IBM MQ MQI client, and is reversible.

Related tasks:

▶ IBM i "Migrating an IBM MQ MQI client to the latest version on IBM i" on page 765
Before migrating an IBM MQ MQI client, create a migration plan. Stop all IBM MQ activity on the client workstation. Upgrade the IBM MQ MQI client installation. Make any essential configuration and application changes.

▶ UNIX ▶ Linux "Migrating an IBM MQ MQI client on UNIX and Linux" on page 736
Before migrating an IBM MQ MQI client, create a migration plan. Stop all IBM MQ activity on the client workstation. Upgrade the IBM MQ MQI client installation. Make any essential configuration and application changes.

Migrating an IBM MQ MQI client to a later version on Windows: ▶ Windows

Before migrating an IBM MQ MQI client on Windows platforms, create a migration plan. Stop all IBM MQ activity on the client workstation. Upgrade the IBM MQ MQI client installation. Make any essential configuration and application changes.

Before you begin

Before starting to migrate a client, create a migration plan. For guidance on what to include in the plan, see "Planning to migrate to a later version on Windows" on page 685.

About this task

IBM MQ MQI client migration is the process of converting IBM MQ MQI client configurations, and client and server channels from one version to another. Client migration is reversible. It is optional and manual on a client workstation and is required and automatic on the IBM MQ server.

You must upgrade an IBM MQ MQI client before migrating a client workstation to make use of new configuration options. You can make configuration changes to client and server connection channels on the server, but they have no effect on a client workstation until the client is upgraded.

Procedure

1. Review the IBM MQ system requirements for the later version of the product. See System Requirements for IBM MQ.
2. Review all the changes in IBM MQ that affect you. For more information, see "Changes that affect migration" on page 626.
3. End all IBM MQ activity on the workstation.

4. Upgrade the client. Select the appropriate option for your enterprise.
   - For a client installation on a workstation, see Installing an IBM MQ client on Windows.
   - For a client installation on an IBM MQ server, see Installing IBM MQ clients and servers on the same system.

What to do next

After upgrading the IBM MQ MQI client, you must check the client channel configuration, and verify that your IBM MQ MQI client applications work correctly with the later version of the product.

Related concepts:

"IBM MQ MQI client migration" on page 650
IBM MQ MQI client migration is the process of converting IBM MQ MQI client configurations, and client and server channels from one version to another. Client migration can take place after upgrading the IBM MQ MQI client, and is reversible.

Related tasks:

"Planning to migrate to a later version on Windows" on page 685
Before migrating from one version of IBM MQ to another on Windows, read this topic, then create your own migration plan based on the outline in this topic.

Restoring an IBM MQ MQI client to an earlier version on Windows: Windows

If you revert an IBM MQ MQI client from a later version of the product to an earlier version of the product, you must undo the configuration changes manually.

About this task

It is unusual to revert earlier IBM MQ MQI client libraries to a workstation. The principal tasks are listed in the following steps.

Procedure

1. End all IBM MQ activity on the workstation.
2. Uninstall the later version of the IBM MQ MQI client code.
3. Follow the client installation procedure for the platform to install the earlier version of the IBM MQ MQI client code.
4. If you configured a Client Connection Definition Table (CCDT) for a queue manager on a later version of the product, revert to using a table created by a queue manager on the earlier version. The CCDT must always be created by a queue manager on the same, or earlier, release to the client.

Migrating IBM MQ library loading to a later version on Windows

Windows

On Windows, no change in the way IBM MQ libraries are loaded is normally required if you upgrade from an earlier version of the product to the later version by replacing an earlier version of the product with the later version, based on the single stage scenario. However, if you choose to take advantage of multi-installation in the later version of the product, based on the side-by-side or multi-stage migration scenarios, you might have to configure the runtime environment differently, for the operating system to load the later version of the IBM MQ library.

Before you begin

To migrate applications from an earlier version of the product to the later version, you must know how the operating system loads a IBM MQ library for an application. Is the load path fixed by the application, and can you set the path in an environment variable? It is not essential to know the name of the IBM MQ library that the application loads. The library name does not change from an earlier version of the product to the later version, although the contents of the library do.

Read “Multi-installation queue manager coexistence on UNIX, Linux, and Windows” on page 668 before starting this task.

Plan and install the later version of IBM MQ, and remember the installation name and whether the installation was set to primary.

About this task

To migrate an application from an earlier version of the product to the later version, you do not have to recompile or relink the application, because the IBM MQ libraries are compatible with later versions; see “Application compatibility and interoperability with later versions of IBM MQ” on page 678.

Windows searches numerous directories for load libraries, called DLLs; see Dynamic-Link Library Search Order. The build procedure for IBM WebSphere MQ Version 7.0.1 or later applications places the IBM MQ libraries to load before any other product libraries in the **cl** command. The IBM MQ .lib libraries must be in the PATH environment variable that you have specified at build time, and the DLL libraries at run time. The PATH variable is used by the application process to find the libraries it must load.

If you have followed this build procedure in the earlier release, then the effect of installing the later version of the product on the libraries that are loaded depends on which migration scenario that you are following:

Single-stage scenario
If you are replacing an earlier version of the product with the later version, based on the single stage scenario described in “Migrating on Windows: single stage” on page 694, you do not, in most cases, need to make any changes to the way IBM MQ libraries are loaded. The possible exception to this is if you changed the location of the libraries from the earlier version, or created symbolic links to the libraries.

Side-by-side and Multi-stage scenarios
If you have chosen a multi-installation approach to installing the later version of the product, based either on the side-by-side scenario described in “Migrating on Windows: side-by-side” on page 699, or the multi-stage migration scenario described in “Migrating on Windows: multi-stage” on page 701, you must investigate whether applications connecting to the later version of the product are linked to, and load libraries from, the correct installation and then modify the environment for the operating system to resolve IBM MQ dependencies for an

application as appropriate. Typically, you can modify the runtime environment, rather than relink the application. You can use the following two commands to assist you in configuring the runtime environment:

- **setmqinst** sets the primary installation; see setmqinst.
- **setmqenv** initializes the command environment by setting environment variables; see setmqenv.

Table 84 summarizes the actions needed for each of these scenarios.

Table 84. Windows configurations

Action / Scenario	**Latest version replaces earlier version in the same location** **Single-stage**	**Latest version replaces earlier version in a different location** **Side-by-side**	**Latest version alongside earlier version** **Multi-stage**
setmqinst	**setmqinst** makes the later version installation primary. The global PATH is changed to point to the later version library and all Windows features work with the later version [See note].		No. The later version installation can be primary, because an earlier version is installed.
No other configuration actions	Library loading works correctly. The global PATH contains the location of the later version libraries. Even if the later version installation is not primary, library loading works correctly. The later version libraries are in the same location as the earlier version libraries were.	Library loading probably works correctly. The library loading might not work, if the application process locally modified the PATH to reference the location of the earlier version libraries. A local setting of PATH might override the global PATH that is set by **setmqinst**.	The library loading continues to work with the earlier version correctly, nothing works with the later version.
setmqenv	Library loading works correctly. **setmqenv** sets the local PATH correctly.		Library loading works correctly, both for the earlier version and the later version. **setmqenv** sets the local PATH correctly for the later version. But the Windows features that depend on the global path do not work correctly with the later version [See note]. The correct earlier version is loaded, because the later version library loads the earlier version library for queue managers that have not been migrated from the earlier version.

Procedure

1. Consider which of the following questions apply to your configuration.
   - Did you follow the build procedure documented in the product documentation for the earlier version of the product? You might be following a different build procedure tailored to your development environment, or adapted from a development tool such as Microsoft Visual Studio.
   - How did you specify the load path for the earlier version?

- Is the application is loaded by another environment, such as Eclipse, or an application server? You must modify the parameters that govern how the parent environment loads applications, not the way the parent environment is loaded.
- Do the functions performed by an application require that the queue manager it connects to is associated with the primary installation?
- What constraints and requirements do you have on how the load path is specified in the later version? Security rules might restrict the use of `LD_LIBRARY_PATH`.
- Is the later version of the product installed alongside the earlier version? If Version 7.0.1 is installed:
  - You cannot make a later installation primary.
  - You cannot install the later version in the default installation path, that was referenced by applications in Version 7.0.1.

2. Identify the installation of the later version of the product, from which the operating system is going to load IBM MQ libraries:
   - If you have a multiple installations of the later versions to load from a server, IBM MQ checks that the installation the library was loaded from is the installation that is associated with any queue manager the application calls. IBM MQ loads the correct library if the wrong library is loaded. It is necessary to configure only one runtime environment for all IBM MQ applications.
   - A typical choice is set the primary installation. Setting an installation to be primary places its library path in the global `PATH` variable.
   - If you upgraded an earlier version installation to the later version, a link path to the earlier version installation now points to an installation containing the later version. Applications that have a fixed linkage path to the earlier version installation now load the libraries for the later installation. They are then switched to the installation that is associated with any queue manager they connect to.
   - If you rebuild an application, it must link to an installation of the later version.
   - If an application uses COM or ActiveX it can connect to any queue manager as long as there is a primary installation and it is Version 7.1 or later.

     Note: If an earlier version of the product is installed, COM or ActiveX server applications connect to queue managers associated only with the earlier installation. COM or ActiveX client applications are not affected by the limitation.
   - If you are running the IBM MQ.NET monitor in transactional mode, the queue manager it connects to must be the primary installation.

What to do next

If you add further installations of the later version of the product, you must decide which installation to make primary, if you have chosen to make any primary. As long as applications load IBM MQ libraries from one of the later version installations, such as the primary installation, they can connect to queue managers associated with any other later version installation.

On Windows, you might build applications with different development tools. You must identify the property of the development tool that sets the `PATH` of the application that is being built, and not the properties of the tool itself. For example, if you are debugging with Microsoft Visual Studio, you can insert a call to **`setmqenv`** in the **`Environment`** property of the debugging section of the **`Configuration`** properties of a project.

A Windows application might call LoadLibrary and specify an explicit load path. You might build a side-by-side assembly and configure an explicit load path. If an application uses either of these mechanisms, and the later version IBM MQ library is not on the same path as the earlier release, you must recompile, or configure and relink your application to load the later version libraries.

Related tasks:

"Migrating IBM MQ library loading to a later version on UNIX and Linux" on page 739
On UNIX and Linux, no change in the way IBM MQ libraries are loaded is normally required if you upgrade from an earlier version of the product to a later version by replacing an earlier version of the product with the later version, based on the single stage scenario. However, if you choose to take advantage of multi-installation in the later version of the product, based on the side-by-side or multi-stage migration scenarios, you might have to configure the runtime environment differently, for the operating system to load the later version of the IBM MQ library.

Related reference:

"Coexistence" on page 664
Queue managers, with different names, can coexist on any server as long as they use the same IBM MQ installation. On z/OS, UNIX, Linux, and Windows, different queue managers can coexist on the same server and be associated with different installations. In addition to queue managers coexisting on a server, objects, and commands must work correctly with different queue managers running at different command levels.

Related information:

Changing the primary installation

Connecting applications in a multiple installation environment

setmqenv

setmqinst

setmqm

Features that can be used only with the primary installation on Windows

Migrating from a single instance to a multi-instance queue manager on Windows

▶ Windows

To migrate 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 and support statement for IBM MQ multi-instance queue managers. 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 **amqmfsck** 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 **amqmfsck** 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 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 (Version 7.0.1 onwards). 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`.

   If you are using an NFS v4 file server, add the line `/MQHA * rw,sync,no_wdelay,fsid=0)` to `etc/exports`, and then start the NFS daemon: `/etc/init.d/nfs start`.
4. Copy the queue manager data and the logs to the share. You can choose to copy files manually, by following the procedure to back up the queue manager. On Windows, you can run the **hamvmqm** command to move the queue manager data to the share. The **hamvmqm** command works for queue managers created before Version 7.0.1, and not reconfigured with a datapath, or for queue managers that do not have a **DataPath** configuration attribute. Choose one of these methods:
   - Follow the instructions in Backing up queue manager data, copying the queue manager data to the share. You must use this method if the **DataPath** configuration attribute is specified for this queue manager.
   - Stop the queue manager, and then type the following command:
     ```
     hamvmqm /m /dd share\data /dd share\logs
     ```
     where *share* is to be the location of the data and logs that you created in step 3.
5. Update the queue manager configuration information stored on the current queue manager server.
   - If you moved the queue manager data and logs by running the **hamvmqm** command, the command has already modified the configuration information correctly for you.
   - If you moved the queue manager data and logs manually, you must complete the following steps.
     a. Modify the log registry key:
     ```
     HKEY_LOCAL_MACHINE\SOFTWARE\IBM\WebSphere MQ\Installation\MQ_INSTALLATION_NAME\Configuration\QueueManager\QMgr
     "LogPath"="share\\logs\\QMgrName\\"
     ```
     b. Modify the Prefix registry key:
     ```
     HKEY_LOCAL_MACHINE\SOFTWARE\IBM\WebSphere MQ\Installation\MQ_INSTALLATION_NAME\Configuration\QueueManager\QMgr
     "Prefix"="share\\data"
     ```

where *QMgrName* is the representation of the queue manager name in the existing registry key on Windows. *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 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, 2009.  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 information:
amqmfsck (file system check)
Application recovery
Automatic client reconnection
Backing up queue manager data
Channel and client reconnection
Changing configuration information on UNIX, Linux, and Windows systems
Moving a queue manager to MSCS storage
Multi-instance queue managers
Queue manager configuration files, qm.ini
Shared file system

Testing a shared file system for compatibility with IBM MQ Multi-instance Queue Managers

Testing and support statement for IBM MQ multi-instance queue managers
Verifying shared file system locking
Windows domains and multi-instance queue managers
Working with services

Reverting to a single-instance queue manager on Windows

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 "Migrating from a single instance to a multi-instance queue manager on Windows" on page 712 to create a multi-instance queue manager.

Migrating MQ Telemetry on Windows

Windows

Follow these instructions to migrate your existing installation of MQ Telemetry to a later version of the product on Windows.

Before you begin

Before proceeding with this task, ensure that you back up your existing IBM MQ installation. You must stop the MQ Telemetry service `SYSTEM.MQXR.SERVICE` before migrating.

About this task

From Version 7.1 onwards, the telemetry server is included in the product as an optional installation.

For Version 7.1 and Version 7.5, the Client Software Development Kit (the telemetry clients) is also included in the optional installation. From Version 8.0 onwards, the Client Software Development Kit is no longer supplied as part of the product. Instead, the current version of the SDK is available as the IBM Messaging Telemetry Clients SupportPac.

Because MQ Telemetry is a component of IBM WebSphere MQ Version 7.1 and later, MQ Telemetry can either be installed with the main product, or installed after the main product has been installed. When you upgrade from a previous version of the product, you must download and use the latest version of the Client Software Development Kit.

After the successful upgrade, Windows systems retain the telemetry data in the installation directory of the product, for example: `C:\Program Files (x86)\IBM\WebSphere MQ`. Telemetry data is migrated to the later version of the product when the queue manager is started again.

Procedure

1. Create a migration plan. See "Planning to migrate to a later version on Windows" on page 685.
2. Migrate your queue managers to the later release.
3. Install MQ Telemetry.
4. Verify that the MQ Telemetry installation was successful. See Verifying the installation of MQ Telemetry.

Results

Message AMQ4616 indicates completion of the task. The existing MQTT channels and previous subscriptions are still present.

Related information:
Installing IBM MQ - overview
Installing MQ Telemetry
Verifying the installation of MQ Telemetry
Verifying the installation of MQ Telemetry by using IBM MQ Explorer

Migrating an MSCS configuration on Windows

Windows

Migrate queue managers in a Microsoft Cluster Service (MSCS) configuration one node at a time, following these instructions.

About this task

These steps are required for a rolling upgrade with a minimum amount of downtime. You must always upgrade an offline node with no online IBM MQ resources. In an Active/Passive configuration, if the node is Passive, you must ensure it cannot be switched to Active during the upgrade process.

The example, "Migrating a four-node MSCS cluster from an earlier version of the product to the latest version," shows this procedure applied to a four-node cluster.

Procedure

1. Modify the possible owners of the IBM MQ resource to encompass only the Active node or nodes. With no owners assigned to Passive nodes, the IBM MQ resource that is being migrated cannot be activated.
2. Ensure that the group containing the IBM MQ resource is currently on one of the nodes defined as a possible owner. The group must include any applications connecting to the queue manager resource.
3. Stop the cluster service on the node being migrated. The MSCS cache is cleared of any IBM MQ DLLs that have been registered.
4. Migrate the selected node by following the standard instructions in "Migrating a queue manager to a later version on Windows" on page 691. Apply the required maintenance level.
5. Start the cluster service on the selected node.
6. On the next node to be migrated, ensure that the IBM MQ resources are offline.
7. Remove this node from the list of possible owners. For clusters with more than two nodes, see the Additional considerations later in this topic.
8. Move the group containing the IBM MQ resource to one of the possible owners and bring it online.
9. Repeat steps 3-8 as necessary for any remaining nodes.

Migrating a four-node MSCS cluster from an earlier version of the product to the latest version

The example in Table 85 on page 718 illustrates the steps involved in migrating a four-node MSCS cluster.

In the example IBM MQ resources include queue managers, applications, and dependant MSCS resources, such as an IP address defined an as MSCS resource. In each step, the changes are italicized.

Step 1 Select the node to migrate and prepare it for upgrading from an earlier version of the product to the latest version.

1. Select node 1 to be migrated and convert it into a Passive node with no running IBM MQ resources.

2. Modify the possible owners of the group containing the IBM MQ resources, to encompass only the required online nodes. Failover does not attempt to switch IBM MQ resources to the node that is not a possible owner. It is safe to migrate that node.
3. Move the group containing the IBM MQ resource to one of the nodes that is a possible owner, and bring it online.
4. Stop the cluster service on the node being migrated. Stopping the service clears the MSCS cache of any IBM MQ libraries that have been registered for MSCS. The node goes offline.

Step 2 Migrate IBM MQ from an earlier version of the product to the latest version

Step 3 Start the cluster service on the selected node. The node becomes online, but it is not a possible owner, so no work is switched to it.

Step 4 Repeat steps 1 - 3 for node 2. Nodes 1 and 2 are now online, and you have migrated them to the latest version. They are still doing no work, as they are not possible owners of any of the IBM MQ resource groups.

Step 5 Migrate the cluster from running an earlier version of the product to the latest version. The number of migrated nodes is now greater or equal to the number of unmigrated nodes.

1. Change the set of possible owners from `3,4` to `1,2`.
2. Move the IBM MQ resource groups from nodes `3` and `4` to nodes `1` and `2` and bring online.
3. From this point onward, the list of possible owners must include migrated nodes only. The IBM MQ resource must never failover to a node running a back level version of the product.

Note: If you must revert IBM MQ to an earlier version, the IBM MQ resources must be removed from MSCS control, before performing an uninstallation of IBM MQ

Step 6 Migrate node 3 to the latest version.

1. Follow steps 1 - 3 for node 3.
2. Add node 3 to the list of possible owners.
3. Move the QMC resource group back from node 1 to node 3 and bring online again.

Step 7 Repeat step 6 for node 4.

Table 85. Migrating a four-node MSCS cluster

Steps		**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**
Node 1	State	Online	*Offline*	Offline	*Online*	Online	Online	Online	Online
	Version	Earlier version	Earlier version	*Latest version*	Latest version	Latest version	Latest version	Latest version	Latest version
	Groups	QMA					*QMC, QMA*	*QMA*	QMA
Node 2	State	Online	Online	Online	Online	Online	Online	Online	Online
	Version	Earlier version	Earlier version	Earlier version	Earlier version	*Latest version*	Latest version	Latest version	Latest version
	Groups	QMB	QMB	QMB	QMB		*QMD, QMB*	QMD, QMB	*QMB*
Node 3	State	Online	Online	Online	Online	Online	Online	Online	Online
	Version	Earlier version	Earlier version	Earlier version	Earlier version	Earlier version	Earlier version	*Latest version*	Latest version
	Groups	QMC	QMC, QMA	QMC, QMA	QMC, QMA	QMC, QMA		*QMC*	QMC

Table 85. Migrating a four-node MSCS cluster (continued)

Steps		0	1	2	3	4	5	6	7
Node 4	State	Online	Online	Online	Online	Online	Online	Online	Online
	Version	Earlier version	Earlier version	Earlier version	Earlier version	Earlier version	Earlier version	Earlier version	*Latest version*
	Groups	QMD	QMD	QMD	QMD	QMD, QMB			*QMD*
Possible Owners		1,2,3,4	*2,3,4*	2,3,4	2,3,4	*3,4*	*1,2*	*1,2,3*	*1,2,3,4*
Task			Update 1			Update 2	Transfer	Update 3	Update 4

What to do next

Additional considerations in an MSCS setup with more than 2 nodes: A cluster might contain enough nodes for you to form a group of migrated queue managers and a group of unmigrated nodes. Switch to the migrated group when it contains half the number of queue managers. Before you have reached the half way point, the unmigrated group are possible owners. When you reach the half way point, switch the possible owners to the migrated group.

Related tasks:

"Migrating a queue manager in a high-availability configuration" on page 814
High-availability configurations of queue managers can increase the availability of IBM MQ applications. If a queue manager, or server fails, it is restarted automatically on another server. You can arrange for IBM MQ MQI client applications to automatically reconnect to the queue manager. Server applications can be configured to start when the queue manager starts.

Related reference:

"MSCS restriction with multiple installations on Windows" on page 641
When you install or upgrade to Version 7.1 or later, the first installation of the product on the server is the only one that can be used with Microsoft Cluster Server (MSCS). No other installations on the server can be used with MSCS. This restriction limits the use of MSCS with multiple installations of the product.

Migrating logs to an Advanced Format disk on Windows

Windows V 9.0.4

An Advanced Format disk is one that has 4096 bytes per sector. The following is applicable only to the Windows platform as Advanced Format disks can be used on other platforms, without carrying out a migration procedure.

Attention: On Windows prior to Version 9.0.4, IBM MQ does not support Advanced Format disks

Note the following:

- A migrated log can be used on any disk whether or not it is Advanced Format.
- If you are not using an Advanced Format disk, you do not need to migrate the log of your queue manager.
- Queue managers that are created at IBM MQ Version 9.0.4 can be used on an Advanced Format disk without being migrated.
- If you use a queue manager that was created before Version 9.0.4 on a native Advanced Format disk, without migrating the queue manager first, the queue manager will not start
- It is possible to start a queue manager on an Advanced Format disk in emulation mode without migration. However IBM MQ log writes will not be on 4k boundaries and so the queue manager will not have data integrity. Once the logs have been migrated, an Advanced Format disk in emulation mode is reliable.

- If you are not sure whether your disk is Advanced Format or not, use the Windows utility **fsutil** to find out.
- The Advanced Format disks that require you to migrate your log, include 4k native disks and 512-byte Emulation disks.
- Using **migmqlog** to change from linear logging to circular logging, or from circular logging to linear logging, also migrates the log so that the log can be used on an Advanced Format disk.

Related tasks:

"Migrating logs on UNIX, Linux, and Windows" on page 820
From IBM MQ Version 9.0.4 you can migrate a circular log to a linear log, or from a linear log to a circular log.

Migrating IBM MQ on UNIX and Linux

UNIX Linux

Migration tasks associated with UNIX and Linux platforms are grouped in this section.

Related concepts:

"Migration concepts and methods" on page 645
An overview of the various concepts and methods for migrating from one release of the product to another.

Related tasks:

IBM i "Migrating IBM MQ on IBM i" on page 749
IBM MQ migration tasks associated with IBM i are grouped in this section.

Windows "Migrating IBM MQ on Windows" on page 684
IBM MQ migration tasks associated with Windows platforms are grouped in this section.

z/OS "Migrating IBM MQ on z/OS" on page 770
Migration tasks associated with z/OS are grouped in this section.

Related reference:

"Changes that affect migration" on page 626
Changes to the product might affect the migration of a queue manager from an earlier release to the current release of IBM MQ, or affect existing applications or configurations. Review these changes before upgrading queue managers to the latest product version and decide whether you must plan to make changes to existing applications, scripts, and procedures before starting to migrate your systems.

Planning to migrate to a later version on UNIX and Linux

Linux UNIX

Before migrating from one version to another on UNIX and Linux, read this topic. Create your own migration plan based on the outline in the planning topic.

If there are concepts about migration you do not understand, read "Migration concepts and methods" on page 645, and the subsections you need, first.

About this task

Use the following steps as a guide to creating a migration plan.

Procedure

1. Review the IBM MQ system requirements for the later version of the product. See System Requirements for IBM MQ.

2. Decide whether to run the earlier version and the later version of the product on the same server, and also which migration method you want to use. See “Migration methods on IBM MQ for Multiplatforms” on page 653.

   Important: Note that if you require a multi-version installation, and you are using a version of the product prior to IBM WebSphere MQ Version 7.0.1, Fix Pack 6, you must install Version 7.0.1, Fix Pack 6 first before installing the later version.
3. Review all the changes in IBM MQ that affect you. For more information, see “Changes that affect migration” on page 626.
4. Review performance changes. See IBM MQ Family - Performance Reports.
5. Review the readme file for the product that you are working with. See IBM MQ, WebSphere MQ, and MQSeries product readmes.
6. Plan the sequence and timing of queue manager upgrades.
   - If the queue manager is part of a queue manager cluster, you must migrate the queue managers that are full repositories first.
   - If the queue manager is part of a high availability cluster, plan the migration to minimize downtime and maximize availability; see “Migrating a queue manager in a high-availability configuration” on page 814.
7. Plan to migrate your queue manager to the later version. See “Migrating a queue manager to a later version on UNIX and Linux” on page 722.

   Backing up queue manager data is part of the queue manager migration task. An alternative approach to backing up queue manager data, is to install and configure a new server. Test the later version with a new queue manager on the new server. When you are ready to go into production on the later version, copy the queue manager configuration and data, to the new server.
8. Plan to update any manual or automated procedures you have written with changes to messages and codes.
9. Decide on what regression tests to perform before putting the queue manager into production on the later version. Include the procedures and applications you identified in steps 6 and 7 in your regression tests.
10. Plan to upgrade your IBM MQ MQI client installations to the later version.
11. Plan to upgrade your client and server applications to use new functions in the later version.

Migrating a queue manager on UNIX and Linux

UNIX Linux

The procedures for migrating a queue manager to a later version of the product, and for restoring a queue manager to an earlier version of the product are detailed in this section.

Related tasks:

IBM i "Migrating a queue manager to a later version on IBM i" on page 751
Follow these instructions to migrate a queue manager from an earlier release to a later release.

Windows "Migrating a queue manager on Windows" on page 691
The procedures for migrating a queue manager to a later version of the product, and for restoring a queue manager to an earlier version of the product are detailed in this section.

z/OS "Migrating IBM MQ on z/OS" on page 770
Migration tasks associated with z/OS are grouped in this section.

Migrating a queue manager to a later version on UNIX and Linux: UNIX Linux

On UNIX and Linux, follow these instructions to migrate a queue manager from an earlier version to a later version of IBM MQ.

Before you begin

If you have installed early support program code on the server, you must delete all the queue managers created with the installation. Uninstall the code before proceeding with installing the production level code.

1. The upgrade from the earlier version to the latesr version of the product requires a full migration of queue managers. Create a migration plan. Use the planning task, "Planning to migrate to a later version on UNIX and Linux" on page 720, as a guide.
2. Review the IBM MQ system requirements for the later version; see System Requirements for IBM MQ.
3. Back up your system before you install a later version of IBM MQ over an earlier version. Once you have started a queue manager you cannot revert to the previous version. If you must restore the system, you cannot recover any work, such as changes to messages and objects, performed by the later version of IBM MQ. For more information about backing up your system, see Backing up and restoring IBM MQ queue manager data.
4. Review any other installed SupportPacs for their applicability to the later version.
5. If you are running on a server with multiple IBM MQ installations, you must identify the installation. Make sure that the commands you enter run against the correct installation; see setmqenv.

About this task

To run a command, the operating system must find the command in an IBM MQ installation. For some commands, you must run the command from the installation that is associated with the correct queue manager. IBM MQ does not switch commands to the correct installation. For other commands, such as **`setmqinst`**, you can run the command from any installation that has the latest version of the product installed.

If an earlier version of the product is installed, the command that is run is the command for that version, unless the search path is overridden by a local setting. You can override the search path by running **`setmqenv`**. If Version 7.0.1 is not installed, you must set the correct path to run a command. If you have set a primary installation, the command that is run is the copy in the primary installation, unless you override the selection with a local search path.

Procedure

1. Log in as a user in `group mqm`.
2. Stop all applications using the IBM MQ installation.

   If you use the Managed File Transfer (MFT) component, ensure that any MFT agents have finished all of the file transfers that they were engaged in. There should be no incomplete transfers associated with the agents, and their SYSTEM.FTE.STATE queues should contain no messages.

3. End all the activity of queue managers associated with the IBM MQ installation.
   a. Run the **dspmq** command to list the state of all the queue managers on the system.

      Run either of the following commands from the installation that you are updating:

      ```
      dspmq -o installation -o status
      dspmq -a
      ```

      dspmq -o installation -o status displays the installation name and status of queue managers associated with all installations of IBM MQ.

      dspmq -a displays the status of active queue managers associated with the installation from which the command is run.
   b. Use the MQSC command **DISPLAY LSSTATUS** to list the status of listeners associated with a queue manager, as shown in the following example:

      ```
      echo "DISPLAY LSSTATUS(*) STATUS" | runmqsc QmgrName
      ```
   c. Run the **endmqm** command to stop each running queue manager associated with this installation.

      ```
                    ┌─-c─┐
      ►►──endmqm────┼─-w─┼──QmgrName──────────────────────────────────────►◄
                    ├─-i─┤
                    └─-p─┘
      ```

      The **endmqm** command informs an application that the queue manager it is connected to is stopping; see Stopping a queue manager.

      For the maintenance to proceed, applications must respond to an **endmqm** command by disconnecting from the queue manager and releasing any IBM MQ libraries they have loaded. If they do not, you must find another way to force applications to release IBM MQ resources, such as by stopping the applications.

      You must also stop applications that are using the client libraries that are part of the installation. Client applications might be connected to a different queue manager, running a different installation of IBM MQ. The application is not informed about queue managers in the current installation being shut down.

      Any applications that continue to have IBM MQ shared libraries from the installation loaded prevent you applying IBM MQ maintenance. An application might disconnect from a queue manager, or be forcibly disconnected, but keep an IBM MQ shared library loaded.

      Note: "Applying maintenance level updates to multi-instance queue managers on UNIX and Linux" on page 611 describes how to apply maintenance to a multi-instance queue manager. A multi-instance queue manager can continue to run on one server, while maintenance is applied to another server.
   d. Stop any listeners associated with the queue managers, using the command:

      ```
      endmqlsr -m QMgrName
      ```
4. Back up the queue manager. Take copies of all the queue manager's data and log file directories, including all subdirectories, and also the `qm.ini` file. For more information, see Backing up and restoring IBM MQ queue manager data.
5. Log in as root.
6. If you are running IBM WebSphere MQ Version 7.0.1, Fix Pack 6 or later, optionally uninstall the current version of IBM MQ. Note, that you carry out this step only if you are doing a single stage migration; see "Migrating on UNIX and Linux: single-stage" on page 724.
7. If you are running IBM WebSphere MQ Version 7.0.1, Fix Pack 5 or earlier, uninstall the current version of IBM MQ. If you require a side-by-side migration or multistage migration only, you must upgrade to IBM WebSphere MQ Version 7.0.1, Fix Pack 6.
8. Install the later version of the product. See the appropriate topic for the platform that your enterprise uses:

AIX Installing IBM MQ Server on AIX.

HP-UX Installing IBM MQ Server on HP-UX.

Linux Installing IBM MQ Server on Linux.

Solaris Installing IBM MQ Server on Solaris.

9. Move the queue manager to the new IBM MQ installation. You need to carry out this step, only if you are running IBM WebSphere MQ Version 7.0.1, Fix Pack 6 or later, and did not uninstall your current version of the product.

   See "Migrating on UNIX and Linux: side-by side" on page 727 or "Migrating on UNIX and Linux: multi-stage" on page 730 for further information.

10. Start the queue manager.

    `strmqm` *`QmgrName`*

    When you first start a queue manager after migration:

    - Any new attributes for existing objects are set to their default values.
    - Any new default objects are created.
    - Queue manager data is migrated.

    Important: Do not use the `-c` option to start the queue manager, unless you explicitly want to reset or re-create the default system objects.

    You must start IBM MQ before you start any listeners.

Backing up and restoring a queue manager

"The version naming scheme for IBM MQ for Multiplatforms" on page 567
From IBM MQ Version 9.0, releases have a three digit Version, Release, and Modification (VRM) code or a four-digit Version, Release, Maintenance, and Fix (VRMF) level code.

Migrating on UNIX and Linux: single-stage: UNIX Linux

Single-stage migration is the term used to describe replacing the only installation of IBM MQ on a server, with a later release. Single stage migration is also known as *upgrading in place* or *in place upgrade*. Until Version 7.0.1, Fix Pack 6, single-stage was the only migration scenario. Single-stage migration preserves existing scripts and procedures for running IBM MQ the most. With other migration scenarios you might change some scripts and procedures, but you can reduce the effect queue manager migration has on users.

Before you begin

Attention: V 9.0.0 From IBM MQ Version 9.0, the `ccsid_part2.tbl` file replaces the existing `ccsid.tbl` file, used in previous versions of the product, to supply additional CCSID information.

The `ccsid_part2.tbl` file takes precedence over the `ccsid.tbl` file and:
- Allows you to add or modify CCSID entries
- Specify default data conversion
- Specify data for different command levels

The `ccsid_part2.tbl` is applicable to the following platforms only:
- Linux Linux - all versions
- Solaris Solaris
- Windows Windows

If you have added any of your own CCSID information into your existing `ccsid.tbl` file, you should copy this information into the new `ccsid_part2.tbl` file, if you want to take advantage of the new formats in your customizations

You should copy the required information, rather than move the information, so that your existing version of IBM MQ continues to work.

About this task

In the single-stage migration scenario, the installation of the later version of the product replaces an earlier version in the same installation location. It is the same migration process that you might have previously used to upgrade the product prior to IBM WebSphere MQ Version 7.0.1, Fix Pack 6. From Version 7.0.1, Fix Pack 6, this method of migration is termed *single-stage* migration, in contrast to *side-by-side* and *multi-stage* migration.

The advantage of single-stage migration is that it changes the configuration of a queue manager on the earlier version as little as possible. Existing applications switch from loading the libraries from the earlier version, to loading the libraries of the later version, automatically. Queue managers are automatically associated with the installation on the later version. Administrative scripts and procedures are affected as little as possible by setting the installation to be the primary installation. If you set the installation of the later version to be the primary installation, commands such as **`strmqm`** work without providing an explicit path to the command.

You can also migrate a queue manager to a later version of the product on a system where an earlier version has been uninstalled. In this case, the queue manager data must have been retained, or restored from a backup.

Procedure

1. Stop local IBM MQ applications.
2. Stop all the queue managers and listeners.
3. Uninstall any fix packs you have installed from the previous IBM MQ version.
4. Upgrade the earlier version of the product to the later version in the same installation directory.
   - A reason for installing into the same location is to simplify application migration. If you change the installation location, you might remove IBM MQ libraries from an application search path. To migrate an application search path you must modify the application environment, or more rarely, the application itself.
   - UNIX Linux The default installation path is specified as a load path in the IBM MQ build scripts for UNIX and Linux. After installation of the later version, the load libraries of the later version of IBM MQ are in the same location as were the libraries of the earlier version. If you built applications by following the examples in the product documentation for the earlier version, the applications load the correct libraries in the later version.

   a. Decide on an installation naming convention. Give the installation a name of your choosing, or accept the default installation name. For the first installation, the default name is *Installation1*. For the second installation, the name is *Installation2*, and so on.

      AIX On AIX there is no option to set the installation name, *Installation1* is set by default.

   b. Upgrade the earlier version of the product to the later version in place, or uninstall the earlier version, without deleting any queue managers, and install the later version in the same default location. Whether you have to uninstall your previous version of the product depends upon your operating system.

      On the following platforms, you do not have to uninstall a previous version of the product:

      - AIX AIX

- IBM i IBM i, where the process is known as a *slip* installation

  AIX If `mqm.xr.clients` and `mqm.txclient.rte` file sets from earlier versions are installed, you must uninstall these file sets from the earlier versions.

  On the following platforms, you must uninstall the previous version of the product:

  - HP-UX HP-UX
  - Linux Linux
  - Solaris Solaris

5. Optional: Make the later version of the installation the primary installation.
   a. Run the **`setmqinst`** command

      ```
      Inst_1_INSTALLATION_PATH/bin/setmqinst -i -n Inst_1
      ```
   - Make the installation primary to avoid specifying a search path to run IBM MQ commands.
   - If there is a primary installation, UNIX and Linux applications that expect to find the IBM MQ library in `/usr/lib`, find a symbolic link to the library in `/usr/lib/32`[6] . `/usr/lib/32` is normally in the default search path. It is also specified as a load path in the IBM MQ build scripts for UNIX and Linux.
   - It is sufficient to link applications only to `/usr/lib`. With a primary installation of the later version of the product defined on the server, an application can connect to any queue manager associated with any installation on the server. IBM MQ loads the correct library for the application.
6. Start the queue managers and applications.
   a. Optional: Run the **`setmqm`** command to associate the queue managers with `Inst_1`.

      ```
      setmqm -m QM1 -n Inst_1
      setmqm -m QM2 -n Inst_1
      ```

      Notes:
      - The **`setmqm`** step is optional only in the case where migration is from IBM WebSphere MQ Version 7.0.1 to a later release. In this case, the **`strmqm`** command automatically associates the queue manager with its own installation.
      - If you are migrating between any other releases of the product, you must use **`setmqm`** to associate the queue managers with the new installation manually.

   b. Run the **`strmqm`** command to start the queue managers and migrate them to the later version of the product.

      ```
      strmqm QM1
      strmqm QM2
      ```

      At this point, queue manager data is migrated and you cannot revert to a previous release.
   - When an application connects to a queue manager, the operating system searches its load path to load the IBM MQ library. A Version 7.1, or later, library contains code that checks that the queue manager is associated with an installation. If a queue manager is associated with a different installation, IBM MQ loads the correct IBM MQ library for the installation the queue manager is associated with.

What to do next

You cannot reinstall an earlier version of the product on a system that has the latest, or any other, version of IBM MQ installed.

6. `/usr/lib` for 64 bit applications.

Related concepts:

"Queue manager coexistence" on page 664
Queue managers, with different names, can coexist on any server as long as they use the same IBM MQ installation. On z/OS, UNIX, Linux, and Windows, different queue managers can coexist on the same server and be associated with different installations.

"Multi-installation queue manager coexistence on UNIX, Linux, and Windows" on page 668
You can install multiple copies of IBM MQ for UNIX, Linux, and Windows on the same server. The installations must be at Version 7.1 or later, with one exception. One Version 7.0.1 installation, at fix pack level 6, or later, can coexist with multiple Version 7.1, or later installations.

Related tasks:

"Planning to migrate to a later version on Windows" on page 685
Before migrating from one version of IBM MQ to another on Windows, read this topic, then create your own migration plan based on the outline in this topic.

"Migrating a queue manager to a later version on UNIX and Linux" on page 722
On UNIX and Linux, follow these instructions to migrate a queue manager from an earlier version to a later version of IBM MQ.

"Migrating a queue manager to a later version on Windows" on page 691
On Windows platforms, follow these instructions to migrate a queue manager from an earlier version to a later version of IBM MQ.

"Migrating IBM MQ library loading from an earlier version of the product to the latest version" on page 684
No change in the way IBM MQ libraries are loaded is normally required if you upgrade from an earlier version of the product to the latest version. You must have followed the instructions on building IBM MQ applications in the earlier version, and you must replace the earlier version with the latest version of the product. If you choose to take advantage of multi-installation in the latest version of the product, based on the side-by-side or multi-stage migration scenarios, you must modify the environment for the operating system to resolve IBM MQ dependencies for an application. Typically, you can modify the runtime environment, rather than relink the application.

Related information:

Installing IBM MQ server on AIX

Installing IBM MQ server on HP-UX

Installing IBM MQ server on Linux

Installing IBM MQ server on Solaris

Associating a queue manager with an installation

Changing the primary installation

Choosing an installation name

setmqenv

setmqinst

setmqm

Migrating on UNIX and Linux: side-by side: UNIX Linux

Side-by-side migration is the term used to describe installing a later version of IBM MQ alongside an earlier version on the same server. Queue managers remain running during the installation and verification of the later version of IBM MQ. They remain associated with the earlier version of IBM MQ. When you decide to migrate queue managers to the later version of IBM MQ, you stop all queue managers, uninstall the earlier version, and migrate them all to the later version of IBM MQ.

Before you begin

If you are using IBM WebSphere MQ Version 7.0.1, you must ensure that you are running IBM WebSphere MQ Version 7.0.1, Fix Pack 6 or later before installing a later version of the product on the same server. For more information about Version 7.0.1 fix packs, see Recommended Fixes for IBM MQ.

Attention: V 9.0.0 From IBM MQ Version 9.0, the `ccsid_part2.tbl` file replaces the existing `ccsid.tbl` file, used in previous versions of the product, to supply additional CCSID information.

The `ccsid_part2.tbl` file takes precedence over the `ccsid.tbl` file and:
- Allows you to add or modify CCSID entries
- Specify default data conversion
- Specify data for different command levels

The `ccsid_part2.tbl` is applicable to the following platforms only:
- Linux Linux - all versions
- Solaris Solaris
- Windows Windows

If you have added any of your own CCSID information into your existing `ccsid.tbl` file, you should copy this information into the new `ccsid_part2.tbl` file, if you want to take advantage of the new formats in your customizations

You should copy the required information, rather than move the information, so that your existing version of IBM MQ continues to work.

About this task

In the side-by-side migration scenario, you install the later version of IBM MQ alongside queue managers that continue to be associated with Version 7.0.1, or later.

When you are ready to migrate the queue managers, and applications, to the later version:
1. Stop all the queue managers.
2. Uninstall the earlier version of the product.
3. Migrate all the queue managers and applications to the later version.

Procedure
1. Install the later version in a different installation directory from the earlier version.
   a. Decide on an installation naming convention. Give the installation a name of your choosing, or accept the default installation name. For the first installation, the default name is *Installation1*. For the second installation, the name is *Installation2*, and so on.

      AIX On AIX there is no option to set the installation name, *Installation1* is set by default.
   b. Verify the installation.

      Run the installation verification procedures and your own tests.
2. Uninstall the earlier version of the product.

   When uninstalling the earlier product, you must stop all queue managers and applications that have loaded an IBM MQ library on the server. For this reason, you might choose to postpone uninstalling the earlier version of the product until a convenient maintenance window. When an earlier version of the product is not installed on a server, it is sufficient to stop the queue managers and applications that have loaded libraries from the installation that you are uninstalling or updating. It is not necessary to stop applications and queue managers associated with other installations.

   a. Stop all applications that have loaded IBM MQ libraries on the server.
   b. Stop the queue managers and listeners on the server.
   c. Uninstall the earlier version of the product.
      - Stop all local IBM MQ applications
      - If you are migrating from IBM WebSphere MQ Version 7.0.1, stop all your queue managers and listeners, ensuring that you do not delete the queue managers.

        Note: If you are migrating from a release other than IBM WebSphere MQ Version 7.0.1 you do not need to stop all the queue managers at this point.

3. Make the later version of the installation the primary installation.
   a. Run the **`setmqinst`** command

      ```
      Inst_1_INSTALLATION_PATH/bin/setmqinst -i -n Inst_1
      ```

   - Make the installation primary to avoid specifying a search path to run IBM MQ commands.
   - If there is a primary installation, UNIX and Linux applications that expect to find the IBM MQ library in `/usr/lib`, find a symbolic link to the library in `/usr/lib/32`[7] . `/usr/lib/32` is normally in the default search path. It is also specified as a load path in the IBM MQ build scripts for UNIX and Linux.
   - It is sufficient to link applications only to `/usr/lib`. With a primary installation of the later version of the product defined on the server, an application can connect to any queue manager associated with any installation on the server. IBM MQ loads the correct library for the application.

   Use the dspmqinst command to discover the *`Installation name`*, or use the default value `Installation 1`.

   Doing this means that you do not have to specify a search path on IBM MQ commands.

4. Start the queue managers and applications.
   - When an application connects to a queue manager, the operating system searches its load path to load the IBM MQ library [8] . A Version 7.1, or later, library contains code that checks that the queue manager is associated with an installation. If a queue manager is associated with a different installation, IBM MQ loads the correct IBM MQ library for the installation the queue manager is associated with.

   During this process you continue to use queue manager QM2 while you upgrade queue manager QM1 and you use queue manager QM1 while you upgrade QM2.

   Note that each queue manager needs to be stopped in order to be associated with the new installation.

What to do next

You cannot reinstall an earlier version of the product on a system that has the latest, or any other, version of IBM MQ installed.

7. `/usr/lib` for 64 bit applications.

8. On Windows, the IBM MQ library is a DLL. A DLL is sometimes called a load library or a shared library. The entry points to a DLL are defined in a link library, with the file extension `.lib32` or `.lib`. The `.lib` library is linked at build-time and the DLL loaded at runtime.

"Planning to migrate to a later version on Windows" on page 685
Before migrating from one version of IBM MQ to another on Windows, read this topic, then create your own migration plan based on the outline in this topic.

Installing IBM MQ server on AIX

Uninstalling IBM MQ on AIX

Uninstalling IBM MQ on HP-UX

Installing IBM MQ server on HP-UX

Uninstalling IBM MQ on Linux

Installing IBM MQ server on Linux

Uninstalling IBM MQ on Solaris

Installing IBM MQ server on Solaris

Associating a queue manager with an installation

Changing the primary installation

Choosing an installation name

"Queue manager coexistence" on page 664
Queue managers, with different names, can coexist on any server as long as they use the same IBM MQ installation. On z/OS, UNIX, Linux, and Windows, different queue managers can coexist on the same server and be associated with different installations.

"Migrating IBM MQ library loading from an earlier version of the product to the latest version" on page 684
No change in the way IBM MQ libraries are loaded is normally required if you upgrade from an earlier version of the product to the latest version. You must have followed the instructions on building IBM MQ applications in the earlier version, and you must replace the earlier version with the latest version of the product. If you choose to take advantage of multi-installation in the latest version of the product, based on the side-by-side or multi-stage migration scenarios, you must modify the environment for the operating system to resolve IBM MQ dependencies for an application. Typically, you can modify the runtime environment, rather than relink the application.

"Multi-installation queue manager coexistence on UNIX, Linux, and Windows" on page 668
You can install multiple copies of IBM MQ for UNIX, Linux, and Windows on the same server. The installations must be at Version 7.1 or later, with one exception. One Version 7.0.1 installation, at fix pack level 6, or later, can coexist with multiple Version 7.1, or later installations.

setmqenv

setmqinst

setmqm

Migrating on UNIX and Linux: multi-stage: UNIX Linux

Multi-stage migration is the term used to describe running a later version of IBM MQ alongside an earlier version on the same server. After installing the later version alongside the earlier version, you can create new queue managers to verify the installation of the later version, and develop new applications. At the same time, you can migrate queue managers and their associated applications from the earlier version to the later version. By migrating queue managers and applications one-by-one, you can reduce the peak workload on staff managing the migration.

Before you begin

If you are using IBM WebSphere MQ Version 7.0.1, you must ensure that you are running IBM WebSphere MQ Version 7.0.1, Fix Pack 6 or later before installing a later version of the product on the same server. For more information about Version 7.0.1 fix packs, see Recommended Fixes for IBM MQ.

Attention: V 9.0.0 From IBM MQ Version 9.0, the `ccsid_part2.tbl` file replaces the existing `ccsid.tbl` file, used in previous versions of the product, to supply additional CCSID information.

The `ccsid_part2.tbl` file takes precedence over the `ccsid.tbl` file and:
- Allows you to add or modify CCSID entries
- Specify default data conversion
- Specify data for different command levels

The `ccsid_part2.tbl` is applicable to the following platforms only:
- Linux Linux - all versions
- Solaris Solaris
- Windows Windows

If you have added any of your own CCSID information into your existing `ccsid.tbl` file, you should copy this information into the new `ccsid_part2.tbl` file, if you want to take advantage of the new formats in your customizations

You should copy the required information, rather than move the information, so that your existing version of IBM MQ continues to work.

Note:
- If an application uses COM or ActiveX it can connect to any queue manager as long as there is a primary installation and it is Version 7.1 or later.
- If you are running the IBM MQ.NET monitor in transactional mode, the queue manager it connects to must be the primary installation.

You cannot migrate these applications to the later version until you uninstall the earlier version.

About this task

In the multi-stage migration scenario, you install the later version of the product alongside running queue managers that continue to be associated with the earlier version. You can create queue managers and run new applications using the later version installation. When you are ready to start migrating queue managers and applications from the earlier, you can do so, one-by-one. When migration to the later version is complete, you can uninstall the earlier version, and make the later version installation the primary installation.

With the multi-stage approach, until you uninstall the earlier version , you must configure an environment to run applications that connect to a queue manager to the later version. You must also provide a path to run IBM MQ commands. Both these tasks are accomplished with the **`setmqenv`** command.

Note: When you have uninstalled the earlier version, and set the later version as a primary installation, in most circumstances it is not necessary to run the **`setmqenv`** command to run applications. It is still necessary to run **`setmqenv`** to set the environment for commands that connect to a queue manager associated with an installation that is not primary.

Procedure

1. Install the later version in a different installation directory from the earlier version and verify the installation.
   a. Decide on an installation naming convention. Give the installation a name of your choosing, or accept the default installation name. For the first installation, the default name is *Installation1*. For the second installation, the name is *Installation2*, and so on.

AIX On AIX there is no option to set the installation name, *Installation1* is set by default.

   b. Verify the installation.

      Run the installation verification procedures and your own tests.

- You might create new queue managers running the later version, and start to develop new applications before migrating applications from the earlier version.

2. Configure the operating system so that applications load the libraries for the later version of the product.
   a. Migrate queue managers one at a time.

      The first set of applications to load the libraries for the later version of the product are the applications that connect to the first queue manager you are going to migrate.

      It does not matter if those applications also connect to other queue managers on the server. If the applications load the later version libraries, IBM MQ automatically loads the libraries for the earlier version for those applications that connect to that version.

      You can either migrate the operating system environment of all applications, or just the applications that connect to the first queue manager you are going to migrate.
   b. Migrate IBM MQ MQI client applications

      Some of the applications might be running as IBM MQ MQI client applications on another workstation. When you migrate a queue manager, clients connected to it continue to run without loading a client library for the later version.

      You can migrate these clients later, when you need to do so.

      Important: If any IBM MQ MQI client applications are using the library for the earlier version on the server, you must eventually migrate the clients to use the later version of the product before you uninstall the earlier version.
3. Migrate an application to load the new library for the later version:
   - Run **setmqenv** to modify the local path that is searched for IBM MQ libraries.
   - Modify the global search path that is searched for IBM MQ libraries.
   - Relink applications with an additional runtime load path.

     Consult operating system documentation about how to modify the global search path, or include a fixed runtime load path in the application load module.

   To run **setmqenv** using the `-s` option:

   ```
   .Inst_1_INSTALLATION_PATH/bin/setmqenv -s -k
   ```

   The `-s` option sets up the environment for the installation that runs the **setmqenv** command.

   The `-k` option inserts the path to the IBM MQ load libraries at the start of the `LD_LIBRARY_PATH` environment variable, and adds the variable to the local environment; see “Loading IBM MQ libraries” on page 670.

   Note: On UNIX the leading `"."` is critical. The dot followed by a space instructs the command shell run **setmqenv** in the same command shell and inherit the environment set by **setmqenv**.
4. Restart the queue manager and the applications that connect to it.
   a. Set up the local environment to the installation `Inst_1`.

      ```
      .Inst_1_INSTALLATION_PATH/bin/setmqenv -s
      ```

      The `-s` option sets up the environment for the installation that runs the **setmqenv** command.
   b. Run the **setmqm** command to associate `QM1` with `Inst_1`.

      ```
      setmqm -m QM1 -n Inst_1
      setmqm -m QM2 -n Inst_1
      ```
   c. Run the **strmqm** command to start `QM1` and migrate it to the later version.

      ```
      strmqm QM1
      strmqm QM2
      ```

   d. Restart application 1

      The application loads the later version library and connects to `QM1`, which is associated with the later version of the product.

5. Migrate all queue managers and applications to the later version.

   Repeat steps 2 on page 732 and 4 on page 732, when required, until all the queue managers and applications are migrated to the later version of the product.

6. Uninstall the earlier version of the product.

   When uninstalling the earlier product, you must stop all queue managers and applications that have loaded an IBM MQ library on the server. For this reason, you might choose to postpone uninstalling the earlier version of the product until a convenient maintenance window. When an earlier version of the product is not installed on a server, it is sufficient to stop the queue managers and applications that have loaded libraries from the installation that you are uninstalling or updating. It is not necessary to stop applications and queue managers associated with other installations.

   a. Stop all applications that have loaded IBM MQ libraries on the server.
   b. Stop the queue managers and listeners on the server.
   c. Uninstall the earlier version of the product.
      - Stop all local IBM MQ applications
      - If you are migrating from IBM WebSphere MQ Version 7.0.1, stop all your queue managers and listeners, ensuring that you do not delete the queue managers.

        Note: If you are migrating from a release other than IBM WebSphere MQ Version 7.0.1 you do not need to stop all the queue managers at this point.

7. Make `Inst_1` the primary installation.

   a. Run the **`setmqinst`** command

      ```
      Inst_1_INSTALLATION_PATH/bin/setmqinst -i -n Inst_1
      ```

   - You do not have to set up a search path to run IBM MQ commands from the primary installation.
   - If you set an installation of the later version of the product as primary on UNIX and Linux, you do not have to set up `LD_LIBRARY_PATH` in most cases. You can remove calls to **`setmqenv`** to set `LD_LIBRARY_PATH`.

What to do next

You cannot reinstall an earlier version of the product on a system that has the latest, or any other, version of IBM MQ installed.

Now that you have uninstalled the earlier version of the product, and made the later installation primary, you can review how the application runtime environment is set. It is no longer necessary to run **`setmqenv`** to set up the search path to load libraries for the later version. If you have only one installation of the later version of the product installed, it is not necessary to run **`setmqenv`** to run commands.

Related concepts:

"Queue manager coexistence" on page 664
Queue managers, with different names, can coexist on any server as long as they use the same IBM MQ installation. On z/OS, UNIX, Linux, and Windows, different queue managers can coexist on the same server and be associated with different installations.

"Multi-installation queue manager coexistence on UNIX, Linux, and Windows" on page 668
You can install multiple copies of IBM MQ for UNIX, Linux, and Windows on the same server. The installations must be at Version 7.1 or later, with one exception. One Version 7.0.1 installation, at fix pack level 6, or later, can coexist with multiple Version 7.1, or later installations.

Related tasks:

"Planning to migrate to a later version on Windows" on page 685
Before migrating from one version of IBM MQ to another on Windows, read this topic, then create your own migration plan based on the outline in this topic.

"Migrating IBM MQ library loading from an earlier version of the product to the latest version" on page 684
No change in the way IBM MQ libraries are loaded is normally required if you upgrade from an earlier version of the product to the latest version. You must have followed the instructions on building IBM MQ applications in the earlier version, and you must replace the earlier version with the latest version of the product. If you choose to take advantage of multi-installation in the latest version of the product, based on the side-by-side or multi-stage migration scenarios, you must modify the environment for the operating system to resolve IBM MQ dependencies for an application. Typically, you can modify the runtime environment, rather than relink the application.

Related information:

Installing IBM MQ server on AIX

Installing IBM MQ server on HP-UX

Installing IBM MQ server on Linux

Installing IBM MQ server on Solaris

Associating a queue manager with an installation

Changing the primary installation

Choosing an installation name

setmqenv

setmqinst

setmqm

Reverting a queue manager to an earlier version on UNIX and Linux: Linux UNIX

On UNIX and Linux, you can revert a queue manager to an earlier version of the product from a later version, if you have made a backup of the system or queue manager. If you have started the queue manager and processed any messages, or changed the configuration, the task cannot give you any guidance on reverting the current state of the queue manager.

Before you begin

1. You must have made a backup of the system or queue manager before you upgraded to the later version. For more information see Backing up and restoring IBM MQ queue manager data
2. If any messages were processed after starting the queue manager, you cannot easily undo the effects of processing the messages. You cannot revert the queue manager to the earlier version of the product in its current state. The task cannot give you any guidance how to deal with subsequent changes that have occurred. For example, messages that were indoubt in a channel, or in a transmission queue on another queue manager, might have been processed. If the queue manager is part of a cluster, then configuration messages and application messages might have been exchanged.

3. If you are running on a server with multiple IBM MQ installations, you must identify the installation. Make sure that the commands you enter run against the correct installation; see setmqenv.

About this task

When you revert to a earlier version of a queue manager, you revert the queue manager to its earlier code level. Queue manager data is reverted to the state it was in when the queue manager was backed up.

Procedure

1. Log in as a user in `group mqm`.
2. Stop all applications using the IBM MQ installation.

   If you use the Managed File Transfer (MFT) component, ensure that any MFT agents have finished all of the file transfers that they were engaged in. There should be no incomplete transfers associated with the agents, and their SYSTEM.FTE.STATE queues should contain no messages.
3. End all the activity of queue managers associated with the IBM MQ installation.
   a. Run the **`dspmq`** command to list the state of all the queue managers on the system.

      Run either of the following commands from the installation that you are updating:
      ```
      dspmq -o installation -o status
      dspmq -a
      ```
      `dspmq -o installation -o status` displays the installation name and status of queue managers associated with all installations of IBM MQ.

      `dspmq -a` displays the status of active queue managers associated with the installation from which the command is run.
   b. Use the MQSC command **`DISPLAY LSSTATUS`** to list the status of listeners associated with a queue manager, as shown in the following example:
      ```
      echo "DISPLAY LSSTATUS(*) STATUS" | runmqsc QmgrName
      ```
   c. Run the **`endmqm`** command to stop each running queue manager associated with this installation.

      ```
                   ┌─-c─┐
      ►►──endmqm───┼─-w─┼──QmgrName──────────────────────────────────────►◄
                   ├─-i─┤
                   └─-p─┘
      ```

      The **`endmqm`** command informs an application that the queue manager it is connected to is stopping; see Stopping a queue manager.

      For the maintenance to proceed, applications must respond to an **`endmqm`** command by disconnecting from the queue manager and releasing any IBM MQ libraries they have loaded. If they do not, you must find another way to force applications to release IBM MQ resources, such as by stopping the applications.

      You must also stop applications that are using the client libraries that are part of the installation. Client applications might be connected to a different queue manager, running a different installation of IBM MQ. The application is not informed about queue managers in the current installation being shut down.

      Any applications that continue to have IBM MQ shared libraries from the installation loaded prevent you applying IBM MQ maintenance. An application might disconnect from a queue manager, or be forcibly disconnected, but keep an IBM MQ shared library loaded.

      Note: The topic, "Applying maintenance level updates to multi-instance queue managers on Windows" on page 577, describes how to apply maintenance to a multi-instance queue manager. A multi-instance queue manager can continue to run on one server, while maintenance is applied to another server.

   d. Stop any listeners associated with the queue managers, using the command:

      ```
      endmqlsr -m QMgrName
      ```
4. Restore the system, or IBM MQ and the queue manager.

   If your backup procedure was to save the queue manager data, you must reinstall IBM MQ:
   a. Uninstall the earlier installation.
   b. Reinstall the product from a manufacturing refresh.
   c. Apply the fix pack and interim fixes that restore IBM MQ to its previous level.
   d. Restore the queue manager data from the backup taken before installing the later version.
5. Restart the earlier version queue manager.

What to do next

You might be reverting to a earlier version on a server with multiple IBM MQ installations. If one of the installations is primary, after reverting the earlier version that installation, by default, becomes the primary installation.

You must review how applications connect to an installation. After reverting to the earlier version, some applications might connect to the wrong installation.

Related information:

Backing up and restoring a queue manager

BFGSS0023E errors and how to avoid them

Migrating an IBM MQ MQI client on UNIX and Linux

UNIX Linux

Before migrating an IBM MQ MQI client, create a migration plan. Stop all IBM MQ activity on the client workstation. Upgrade the IBM MQ MQI client installation. Make any essential configuration and application changes.

Related concepts:

"IBM MQ MQI client migration" on page 650
IBM MQ MQI client migration is the process of converting IBM MQ MQI client configurations, and client and server channels from one version to another. Client migration can take place after upgrading the IBM MQ MQI client, and is reversible.

Related tasks:

IBM i "Migrating an IBM MQ MQI client to the latest version on IBM i" on page 765
Before migrating an IBM MQ MQI client, create a migration plan. Stop all IBM MQ activity on the client workstation. Upgrade the IBM MQ MQI client installation. Make any essential configuration and application changes.

Windows "Migrating an IBM MQ MQI client on Windows" on page 707
Before migrating an IBM MQ MQI client, create a migration plan. Stop all IBM MQ activity on the client workstation. Upgrade the IBM MQ MQI client installation. Make any essential configuration and application changes.

Migrating an IBM MQ MQI client to a later version on UNIX and Linux: Linux UNIX

To upgrade a client to a later version of the product on UNIX and Linux, you must first stop all IBM MQ activity on the workstation, then uninstall the earlier version and install the later version. After you have upgraded the client, you can then make any essential configuration and application changes.

Before you begin

Before migrating an IBM MQ MQI client on UNIX and Linux, first create a migration plan. For guidance on what to include in the plan, see "Planning to migrate to a later version on UNIX and Linux" on page 720, as a guide.

About this task

IBM MQ MQI client migration is the process of converting IBM MQ MQI client configurations, and client and server channels from one version to another. Client migration is reversible. It is optional and manual on a client workstation and is required and automatic on the IBM MQ server.

You must upgrade an IBM MQ MQI client before migrating a client workstation to make use of new configuration options. You can make configuration changes to client and server connection channels on the server, but they have no effect on a client workstation until the client is upgraded.

Procedure

1. Review the IBM MQ system requirements for the later version of the product. See System Requirements for IBM MQ.
2. Review all the changes in IBM MQ that affect you. For more information, see "Changes that affect migration" on page 626.
3. End all IBM MQ activity on the workstation. You are now ready to upgrade the client. Follow the instructions for the appropriate platform that your enterprise uses.
4. AIX To upgrade the client on AIX:
   a. Uninstall your existing IBM MQ client installation. For more information, see Uninstalling or modifying IBM MQ on AIX.
   b. Follow the client installation procedure to install the upgraded version of the IBM MQ client:
      - For a client installation on a workstation, see Installing an IBM MQ client on AIX
      - For a client installation on an IBM MQ server, see Installing IBM MQ clients and servers on the same system.
5. HP-UX To upgrade the client on HP-UX:
   a. Uninstall your existing IBM MQ client installation. For more information, see Uninstalling or modifying IBM MQ on HP-UX.
   b. Follow the client installation procedure to install the upgraded version of the IBM MQ client:
      - For a client installation on a workstation, see Installing an IBM MQ client on HP-UX.
      - For a client installation on an IBM MQ server, see Installing IBM MQ clients and servers on the same system.
6. Linux To upgrade the client on Linux:
   a. Uninstall your existing IBM MQ client installation. For more information, see Uninstalling or modifying IBM MQ on Linux.
   b. Follow the client installation procedure to install the upgraded version of the IBM MQ client:
      - For a client installation on a workstation, see Installing an IBM MQ client on Linux.
      - For a client installation on an IBM MQ server, see Installing IBM MQ clients and servers on the same system.
7. Solaris To upgrade the client on Solaris:
   a. Uninstall your existing IBM MQ client installation. For more information, see Uninstalling IBM MQ on Solaris.
   b. Follow the client installation procedure to install the upgraded version of the IBM MQ client:
      - For a client installation on a workstation, see Installing an IBM MQ client on Solaris.

- For a client installation on an IBM MQ server, see Installing IBM MQ clients and servers on the same system.

What to do next

After upgrading the IBM MQ MQI client, you must check the client channel configuration, and verify that your IBM MQ MQI client applications work correctly with the later version of the product.

Related concepts:

"IBM MQ MQI client migration" on page 650
IBM MQ MQI client migration is the process of converting IBM MQ MQI client configurations, and client and server channels from one version to another. Client migration can take place after upgrading the IBM MQ MQI client, and is reversible.

Related tasks:

"Planning to migrate to a later version on UNIX and Linux" on page 720
Before migrating from one version to another on UNIX and Linux, read this topic. Create your own migration plan based on the outline in the planning topic.

Restoring an IBM MQ MQI client to an earlier version on UNIX and Linux: UNIX Linux

To revert a client to an earlier version of the product on UNIX and Linux, you must uninstall the later version, and then install the earlier version.

About this task

If you revert an IBM MQ MQI client and client connection to an earlier code level, you must undo the configuration changes manually.

It is unusual to revert earlier IBM MQ MQI client libraries to a workstation.

Procedure

1. End all IBM MQ activity on the workstation. You are now ready to restore the client to the earlier version. Follow the instructions for the appropriate platform that your enterprise uses.
2. AIX To revert the client to the earlier version on AIX:
   a. Uninstall the IBM MQ MQI client code for the later version. For more information, see Uninstalling or modifying IBM MQ on AIX.
   b. Follow the client installation procedure to install the IBM MQ MQI client for the earlier version. For more information, see the client installation procedure for the earlier version that you want to install.
3. HP-UX To revert the client to the earlier version on HP-UX:
   a. Uninstall the IBM MQ MQI client code for the later version. For more information, see Uninstalling or modifying IBM MQ on HP-UX.
   b. Follow the client installation procedure to install the IBM MQ MQI client for the earlier version: For more information, see the client installation procedure for the earlier version that you want to install.
4. Linux To revert the client to the earlier version on Linux:
   a. Uninstall the IBM MQ MQI client code for the later version. For more information, see Uninstalling or modifying IBM MQ on Linux.
   b. Follow the client installation procedure to install the IBM MQ MQI client for the earlier version: For more information, see the client installation procedure for the earlier version that you want to install.

5. Solaris To revert the client to the earlier version on Solaris:
   a. Uninstall the IBM MQ MQI client code for the later version. For more information, see Uninstalling IBM MQ on Solaris.
   b. Follow the client installation procedure to install the IBM MQ MQI client for the earlier version. For more information, see the client installation procedure for the earlier version that you want to install.
6. If you configured a Client Connection Definition Table (CCDT) for a queue manager using the later version, revert to using a table created by a queue manager for the earlier version. If a client uses CCDT to connect to a queue manager, the CCDT can be at a version greater than, less than, or equal to that of the client. For more information, see MQI client: Client Channel Definition Table (CCDT).

Migrating IBM MQ library loading to a later version on UNIX and Linux

Linux UNIX

On UNIX and Linux, no change in the way IBM MQ libraries are loaded is normally required if you upgrade from an earlier version of the product to a later version by replacing an earlier version of the product with the later version, based on the single stage scenario. However, if you choose to take advantage of multi-installation in the later version of the product, based on the side-by-side or multi-stage migration scenarios, you might have to configure the runtime environment differently, for the operating system to load the later version of the IBM MQ library.

Before you begin

To migrate applications from an earlier version of the product to the later version, you must know how the operating system loads a IBM MQ library for an application. Is the load path fixed by the application, and can you set the path in an environment variable? It is not essential to know the name of the IBM MQ library that the application loads. The library name does not change from an earlier version of the product to the later version, although the contents of the library do.

Read "Multi-installation queue manager coexistence on UNIX, Linux, and Windows" on page 668 before starting this task.

Plan and install the later version of IBM MQ, and remember the installation name and whether the installation was set to primary.

About this task

To migrate an application from an earlier version of the product to the later version, you do not have to recompile or relink the application, because the IBM MQ libraries are compatible with later versions; see "Application compatibility and interoperability with later versions of IBM MQ" on page 678.

In Version 7.0.1 and later, the build procedure for IBM MQ applications is to include an explicit library path to the location of the IBM MQ libraries, and to `/usr/lib`, in the link step of the compiler, as shown in Figure 75 on page 740. The build procedure is the same for the later version of the product.

```
gcc -m32 -o amqsput_32_r amqsput0.c -I/opt/mqm/inc -L/opt/mqm/lib
-Wl,-rpath=/opt/mqm/lib -Wl,-rpath=/usr/lib -lmqm_r -lpthread
```

Figure 75. Linux C server application, 32 bit, threaded compile and link

The example shown in Figure 75 is for Linux, but the build step for UNIX platforms is similar.

If you have followed this build procedure in the earlier release, then the effect of installing the later version of the product on the libraries that are loaded depends on which migration scenario that you are following:

Single-stage scenario
If you are replacing an earlier version of the product with the later version, based on the single stage scenario described in "Migrating on UNIX and Linux: single-stage" on page 724, you do not, in most cases, need to make any changes to the way IBM MQ libraries are loaded. The possible exception to this is if you changed the location of the libraries from the earlier version, or created symbolic links to the libraries.

Side-by-side and Multi-stage scenarios
If you have chosen a multi-installation approach to installing the later version of the product, based either on the side-by-side scenario described in "Migrating on UNIX and Linux: side-by side" on page 727, or the multi-stage migration scenario described in "Migrating on UNIX and Linux: multi-stage" on page 730, you must investigate whether applications connecting to the later version of the product are linked to, and load libraries from, the correct installation and then modify the environment for the operating system to resolve IBM MQ dependencies for an application as appropriate. Typically, you can modify the runtime environment, rather than relink the application. You can use the following two commands to assist you in configuring the runtime environment:

- **`setmqinst`** sets the primary installation; see setmqinst.
- **`setmqenv`** initializes the command environment by setting environment variables; see setmqenv.

Table 86 summarizes the actions needed for each of these scenarios. The examples in Table 86 are all based on Linux, but the actions for UNIX are similar.

Table 86. UNIX and Linuxconfigurations

Action \ Scenario	Latest version replaces earlier version in the same location Single-stage	Latest version replaces earlier version in a different location Side-by-side	Latest version alongside earlier version Multi-stage
`setmqinst`	**`setmqinst`** makes the later version installation primary. Symbolic links to the IBM MQ link libraries are inserted into `/usr/lib`.		No. The later version installation can be primary, because an earlier version is installed.
No other configuration actions	Library loading works correctly. Library loading works, even without the later version installation being made primary, because the libraries are installed in `/opt/mqm/lib` and the application was built with the link option, `-rpath=/opt/mqm/lib`	Library loading works correctly. Library loading works, because the installation is primary, and the application was built with the link option, `-rpath=/usr/lib`.	The library loading continues to work with the earlier version correctly, nothing works with the later version.

Table 86. UNIX and Linuxconfigurations (continued)

Scenario / Action	Latest version replaces earlier version in the same location Single-stage	Latest version replaces earlier version in a different location Side-by-side	Latest version alongside earlier version Multi-stage
setmqenv, without setting the `-k` or `-l` options.	Library loading works correctly. **setmqenv** is unnecessary. Library loading works, because the libraries are installed in `/opt/mqm/lib` and the application was built with the link option, `-rpath=/opt/mqm/lib`.	Library loading works correctly. **setmqenv** is unnecessary. Library loading works, because the installation is primary, and the application was built with the link option, `-rpath=/usr/lib`.	The library loading continues to work with the earlier version correctly, nothing works with the later version.
setmqenv, with the `-k` or `-l` options set.	Library loading works correctly.		Library loading works correctly, both for the earlier version and the later version. The correct earlier version is loaded, because the later version library loads the earlier version library for queue managers that have not been migrated from the earlier version.
	The operating system finds the IBM MQ library location set by **setmqenv**. **setmqenv** adds the location to `LD_LIBRARY_PATH` (`LIBPATH` on AIX. On HP-UX `LD_LIBRARY_PATH` is set, not `SHLIB_PATH`.) . `LD_LIBRARY_PATH` is searched before paths set in the application or paths in the default search path. Not all applications can load a library using `LD_LIBRARY_PATH`. In which case the application works only if the library location is `/opt/mqm/lib` or `/usr/lib`		

Procedure

1. Consider which of the following questions apply to your configuration.
   - Did you follow the build procedure documented in the product documentation for the earlier version of the product? You might be following a different build procedure tailored to your development environment, or adapted from a development tool.
   - How did you specify the load path for the earlier version?
   - Is the application is loaded by another environment, such as Eclipse, or an application server? You must modify the parameters that govern how the parent environment loads applications, not the way the parent environment is loaded.
   - What constraints and requirements do you have on how the load path is specified in the later version? Security rules might restrict the use of `LD_LIBRARY_PATH`.
   - Is the later version of the product installed alongside the earlier version? If Version 7.0.1 is installed:
     - You cannot make a later installation primary.
     - You cannot install the later version in the default installation path, that was referenced by applications in Version 7.0.1.
2. Identify the installation of the later version of the product, from which the operating system is going to load IBM MQ libraries:
   - If you have a multiple installations of the later versions to load from a server, IBM MQ checks that the installation the library was loaded from is the installation that is associated with any queue manager the application calls. IBM MQ loads the correct library if the wrong library is loaded. It is necessary to configure only one runtime environment for all IBM MQ applications.

- A typical choice is to set the primary installation. Setting an installation to be primary places symbolic links to the IBM MQ libraries in `/usr/lib`. Applications built following the Version 7.0 instructions have an explicit link to `/usr/lib`. `/usr/lib` is also normally in the default library search path.
- If you upgraded an earlier version installation to the later version, a link path to the earlier version installation now points to an installation containing the later version. Applications that have a fixed linkage path to the earlier version installation now load the libraries for the later installation. They are then switched to the installation that is associated with any queue manager they connect to.
- If you rebuild an application, it must link to an installation of the later version.
- AIX If you set `LD_LIBRARY_PATH`, or `LIBPATH` on AIX, you must check that the application is able to use `LD_LIBRARY_PATH`. `setuid` or `setgid`, applications, or applications built in other ways, might ignore `LD_LIBRARY_PATH` for security reasons.

What to do next

If you add further installations of the later version of the product, you must decide which installation to make primary, if you have chosen to make any primary. As long as applications load IBM MQ libraries from one of the later version installations, such as the primary installation, they can connect to queue managers associated with any other later version installation.

Related tasks:

"Migrating IBM MQ library loading to a later version on Windows" on page 709
On Windows, no change in the way IBM MQ libraries are loaded is normally required if you upgrade from an earlier version of the product to the later version by replacing an earlier version of the product with the later version, based on the single stage scenario. However, if you choose to take advantage of multi-installation in the later version of the product, based on the side-by-side or multi-stage migration scenarios, you might have to configure the runtime environment differently, for the operating system to load the later version of the IBM MQ library.

Related reference:

"Coexistence" on page 664
Queue managers, with different names, can coexist on any server as long as they use the same IBM MQ installation. On z/OS, UNIX, Linux, and Windows, different queue managers can coexist on the same server and be associated with different installations. In addition to queue managers coexisting on a server, objects, and commands must work correctly with different queue managers running at different command levels.

Related information:

External library and control command links to primary installation on UNIX and Linux

Connecting applications in a multiple installation environment

Changing the primary installation

setmqenv

setmqinst

setmqm

Loading IBM MQ libraries

Migrating from a single instance to a multi-instance queue manager on UNIX and Linux

Linux UNIX

To migrate a single instance queue manager, to a multi-instance queue manager, on UNIX 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 and support statement for IBM MQ multi-instance queue managers. 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 **`amqmfsck`** 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 **`amqmfsck`** 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 UNIX and Linux, the `uid` and `gid` for `mqm` in `/etc/password` must be the same on each system; see Create 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 (Version 7.0.1 onwards) in Example directory configurations on UNIX. 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 `rwx`.
4. Copy the queue manager data and the logs to the share. You can choose to copy files manually, by following the procedure to back up the queue manager. Choose one of these methods:

- Follow the instructions in Backing up queue manager data, copying the queue manager data to the share. You must use this method if the **DataPath** configuration attribute is specified for this queue manager.
- Stop the queue manager, and then type the command,

  ```
  hamvmqm /m /dd share\data /dd share\logs
  ```

  Where *share* is to be the location of the data and logs that you created in step 3 on page 743.

5. Update the queue manager configuration information stored on the current queue manager server.
   - If you moved the queue manager data and logs by running the **hamvmqm** command, the command has already modified the configuration information correctly for you.
   - If you moved the queue manager data and logs manually, you must complete 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 UNIX 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 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, 2009.  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 information:

amqmfsck (file system check)

Application recovery

Automatic client reconnection

Backing up queue manager data

Channel and client reconnection

Changing configuration information on UNIX, Linux, and Windows systems

Create a multi-instance queue manager on Linux

Moving a queue manager to MSCS storage

Multi-instance queue managers

Queue manager configuration files, qm.ini

Shared file system

Testing a shared file system for compatibility with IBM MQ Multi-instance Queue Managers

Testing and support statement for IBM MQ multi-instance queue managers

The IBM MQ configuration file, mqs.ini

Verifying shared file system locking

Reverting to a single-instance queue manager on UNIX and Linux

Linux UNIX

Revert a multi-instance queue manager to a single instance queue manager, on UNIX 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 "Migrating from a single instance to a multi-instance queue manager on UNIX and Linux" on page 742 to create a multi-instance queue manager.

Cleaning up after using the `rpm` freshen or upgrade options on Linux

▶ Linux

The use of **rpm** upgrade or freshen options is not supported. If you use the options, follow this cleanup procedure, and then install following the correct steps.

Before you begin

You have attempted to upgrade IBM MQ for Linux using `rpm -U` or `rpm -F`

About this task

By using the freshen or upgrade options, you might have deleted your old IBM MQ package entries from the **rpm** database without removing the product from your system. You might also have partially installed IBM MQ

Procedure

Follow these steps to clean up your system.

1. Find out which IBM MQ MQ packages still have entries in your RPM database.
   ```
   rpm -qa | grep MQSeries
   ```
2. Remove all remaining IBM MQ packages from your system.
   ```
   rpm -e package-name
   ```
3. Remove the `/opt/mqm` directory.
   ```
   rm -rf /opt/mqm
   ```

Rebuilding a C++ application on Linux

Linux

C++ IBM MQ MQI client and server applications on Linux must be recompiled using GNU Compiler Collection (GCC) 4.1.2, or later. Compilers older than GCC 4.1.2 are no longer supported. The C++ GCC 4.1.2 run time libraries, or later, must be installed in `/usr/lib` or `/usr/lib64`

If you are using one of the supported Linux distributions, the libraries are correctly installed; see System Requirements for IBM MQ.

The GCC 4.1.2 libraries support SSL and TLS connections from an IBM MQ MQI client. SSL and TLS use GSKit version 8, which depends on `libstdc++.so.6`. `libstdc++.so.6` is included in GCC 4.1.2.

Before you begin

1. Check the required level of GCC for your distribution of Linux; see System Requirements for IBM MQ.
2. If you are using SSL or TLS, also check the required level of `libstdc++.so`.
3. Check whether the application requires rebuilding. Run the following command to display what version of `libstdc++.so` the application depends upon. If the result is less than `libstdc++.so.6`, you must rebuild your application.
   ```
   ldd ApplicationPath
   ```

About this task

The task describes the steps required to rebuild a Linux C++ IBM MQ application. For more detailed instructions about building Linux applications for IBM MQ ; see Building your procedural application on Linux

Procedure

1. Check that the required GCC library is installed correctly.

   Run one of the following commands:
   - Check the 32 bit library on an x86 Linux system:
     ```
     ls -l /usr/lib/libstdc++.so.6
     ```
   - Check the 64 bit library on any other Linux system.
     ```
     ls -l /usr/lib64/libstdc++.so.6
     ```
2. Check that the GCC compiler is at least at version 4.1.2

   Run the following command to display the version of GCC.
   ```
   gcc -v
   ```
3. Rebuild the application

   The commands to compile and link Linux C++ applications are described in Building 32-bit applications and Building 64-bit applications

What to do next

When you deploy your Linux C++ application, ensure that the same GCC runtime library is correctly installed on the run time system.

Migrating MQ Telemetry on Linux

▶ Linux

Follow these instructions to migrate your existing installation of MQ Telemetry on Linux to the latest version of the product.

Before you begin

Before proceeding with this task, ensure that you back up your existing IBM MQ installation. You must stop the MQ Telemetry service `SYSTEM.MQXR.SERVICE` before migrating.

About this task

From Version 7.1 onwards, the telemetry server is included in the product as an optional installation.

For Version 7.1 and Version 7.5, the Client Software Development Kit (the telemetry clients) is also included in the optional installation. From Version 8.0 onwards, the Client Software Development Kit is no longer supplied as part of the product. Instead, the current version of the SDK is available as the IBM Messaging Telemetry Clients SupportPac.

Because MQ Telemetry is a component of IBM WebSphere MQ Version 7.1 and later, MQ Telemetry can either be installed with the main product, or installed after the main product has been installed. When you upgrade from a previous version of the product, you must download and use the latest version of the Client Software Development Kit.

After the successful upgrade, Linux systems retain all telemetry data kept in `/var/mqm`. Telemetry data is migrated to the later version of the product when the queue manager is started again.

Procedure

1. Create a migration plan. See "Planning to migrate to a later version on UNIX and Linux" on page 720.
2. Migrate your queue managers to the latest release.
3. Install MQ Telemetry.
4. Verify that the MQ Telemetry installation was successful. See Verifying the installation of MQ Telemetry.

Results

Message AMQ4616 indicates completion of the task. The existing MQTT channels and previous subscriptions are still present.

Related information:
Installing MQ Telemetry
Verifying the installation of MQ Telemetry
Verifying the installation of MQ Telemetry by using IBM MQ Explorer

Migrating IBM MQ on IBM i

IBM i

IBM MQ migration tasks associated with IBM i are grouped in this section.

Procedure

- For information about creating a migration plan, see "Planning to migrate to the latest version on IBM i."
- For information about migrating an IBM MQ classes for JMS and IBM MQ classes for Java client, see "Migrating an IBM MQ classes for JMS and Java client on IBM i" on page 751.
- For information about migrating a queue manager from a previous release, see "Migrating a queue manager to a later version on IBM i" on page 751 and "Migrating a queue manager to a later version on IBM i - alternative method" on page 762.
- For information about upgrading an IBM MQ system, see "Upgrading an entire IBM MQ system on IBM i" on page 765.
- For information about upgrading an IBM MQ MQI client installation, see "Migrating an IBM MQ MQI client to the latest version on IBM i" on page 765.
- For information about converting a single instance queue manager to a multi-instance queue manager, see "Migrating from a single instance to a multi-instance queue manager on IBM i" on page 766.
- For information about reverting a multi-instance queue manager to a single instance queue manager, see "Reverting to a single-instance queue manager on IBM i" on page 769.

Related concepts:
"Migration concepts and methods" on page 645
An overview of the various concepts and methods for migrating from one release of the product to another.

Related tasks:
UNIX Linux "Migrating IBM MQ on UNIX and Linux" on page 720
Migration tasks associated with UNIX and Linux platforms are grouped in this section.

Windows "Migrating IBM MQ on Windows" on page 684
IBM MQ migration tasks associated with Windows platforms are grouped in this section.

z/OS "Migrating IBM MQ on z/OS" on page 770
Migration tasks associated with z/OS are grouped in this section.

Related reference:
"Changes that affect migration" on page 626
Changes to the product might affect the migration of a queue manager from an earlier release to the current release of IBM MQ, or affect existing applications or configurations. Review these changes before upgrading queue managers to the latest product version and decide whether you must plan to make changes to existing applications, scripts, and procedures before starting to migrate your systems.

Planning to migrate to the latest version on IBM i

IBM i

Before migrating from one version to another on IBM i, read this planning topic, then create your own migration plan based on the outline in this topic.

Before you begin

If there are concepts about migration you do not understand, read "Migration concepts and methods" on page 645, and the subsections you need, first.

About this task

Use the following steps as a guide to creating a migration plan.

Procedure

1. Review the IBM MQ system requirements for the latest version of the product. See System Requirements for IBM MQ.
2. Decide whether to run the current version of your product and the latest version on the same server.
3. Review all the changes in IBM MQ that affect you. For more information, see "Changes that affect migration" on page 626.
4. Review performance changes. See IBM MQ Family - Performance Reports.
5. Review the latest readme file for the product that you are working with. See IBM MQ, WebSphere MQ, and MQSeries product readmes.
6. Plan the sequence and timing of queue manager upgrades.
   - If the queue manager is part of a queue manager cluster, you must migrate the queue managers that are full repositories first.
   - If the queue manager is part of a high availability cluster, plan the migration to minimize downtime and maximize availability; see "Migrating a queue manager in a high-availability configuration" on page 814.
7. Plan to migrate your queue manager to the latest version. See IBM i - Migrating a queue manager from a previous release to the latest release or Migrating a queue manager from a previous release to the latest release, alternative method

   Backing up queue manager data is part of the queue manager migration task. An alternative approach to backing up queue manager data, is to install and configure a new server. Test the latest version with a new queue manager on the new server. When you are ready to go into production on the latest version, copy the queue manager configuration and data, to the new server.
8. Plan to update any manual or automated procedures you have written with changes to messages and codes.
9. Decide on what regression tests to perform before putting the queue manager into production on the latest version. Include the procedures and applications you identified in steps 6 and 7 in your regression tests.
10. Plan to upgrade your IBM MQ MQI client installations to the latest version.
11. Plan to upgrade your client and server applications to use new functions in the latest version.

Migrating an IBM MQ classes for JMS and Java client on IBM i

IBM i

If you have IBM MQ Java SupportPac MA88 installed, you must uninstall it first.

Before you begin

SupportPac MQ88 is installed.
If you try to install the latest version of IBM MQ classes for Java anyway, the installation fails with a warning requesting you to uninstall the old client. You must follow the steps in this task to uninstall IBM MQ classes for Java and IBM MQ classes for JMS.

A previous version of IBM MQ classes for Java is installed.
Installation of the latest version of IBM MQ classes for Java uninstalls the previous version automatically. Do not follow the steps in this task.

About this task

The steps in this task uninstall the IBM MQ classes for JMS and Java.

Procedure

To uninstall the previous IBM MQ Java client:

1. Delete the `QMQMJAVA` library and the `/QIBM/ProdData/mqm/java` directory, by issuing the command:
   ```
   DLTLICPGM LICPGM(5648C60) OPTION(*ALL)
   ```
2. If the previous step failed to delete the IFS directory `/QIBM/ProdData/mqm/java` and its subdirectories, use the **`EDTF`** command, for example:
   ```
   EDTF STMF('/QIBM/ProdData/mqm')
   ```
   and select option 9 against the `java` directory.

Migrating a queue manager to a later version on IBM i

IBM i

Follow these instructions to migrate a queue manager from an earlier release to a later release.

Before you begin

If you decide to do a side-by-side installation, you must prepare the new server first, installing the prerequisite software.

1. Create a migration plan. Use the planning task, Planning migration to the latest version, as a guide.
2. Review the IBM MQ system requirements for the latest release of the product; see System Requirements for IBM MQ
3. Review any other installed SupportPacs for their applicability to the latest release of IBM MQ.

About this task

There are various types of migration:

- The migration takes place on the same machine, optionally accompanied by a hardware upgrade. This migration is referred to as a *slip installation*. On IBM i, uninstalling the earlier version of the product before you install the later version is optional.
- The migration takes place on a different machine. This migration is referred to as a *side-by-side installation*.

A side-by-side installation gives you the option of preparing the new environment first, without interrupting the queue manager. It also gives you the limited option of reverting to use the earlier release installation, if the migration is unsuccessful. It is limited, because you cannot restore the queue manager data from the later version. You must restart processing with the queue manager data at the point you stopped the queue manager on the earlier release.

If you want to add Advanced Message Security to your system, you must select Option (2) when you install the product; see Installing Advanced Message Security on IBM i for further information.

Related tasks:

UNIX Linux "Migrating a queue manager on UNIX and Linux" on page 721
The procedures for migrating a queue manager to a later version of the product, and for restoring a queue manager to an earlier version of the product are detailed in this section.

Windows "Migrating a queue manager on Windows" on page 691
The procedures for migrating a queue manager to a later version of the product, and for restoring a queue manager to an earlier version of the product are detailed in this section.

z/OS "Migrating IBM MQ on z/OS" on page 770
Migration tasks associated with z/OS are grouped in this section.

Installation methods on IBM i: IBM i

Select a slip installation or a side-by-side installation to upgrade IBM MQ for IBM i.

About this task

A slip installation upgrades IBM MQ for IBM i on a computer with an earlier version is installed.

A side-by-side installation upgrades IBM MQ for IBM i on a different computer. You must save your queue managers before you start.

Follow the steps in the following tasks to carry out an upgrade.

The steps for both forms of upgrade are identical, except that you do not carry out the actions described in "Restore queue managers after upgrading IBM MQ on IBM i" on page 760 for a slip installation.

End IBM MQ activity on IBM i: IBM i

End IBM MQ applications and connections, and remove any unwanted or indoubt messages.

About this task

Before performing a slip installation or side-by-side installation, carry out the following procedure:

Procedure

1. Sign on to the system with a user profile that has *ALLOBJ special authority, for example QSECOFR.
2. Stop all applications that are using the existing version of IBM MQ. Use the command `WRKMQM`, option 22, Work with queue manager jobs, to help find them.
3. End all channels for all queue managers on the system. To do so, use the WRKMQMCHL command and select option 15.
4. On each queue manager, end the command server. To do so, enter the command:
   ```
   ENDMQMCSVR MQMNAME( QMGRNAME ) OPTION(*IMMED)
   ```
   where *QMGRNAME* is the name of the queue manager.

5. Remove any unwanted messages from your queues.
6. Resolve any in-doubt messages that are held by sender or server channels. To do so, use the WRKMQMCHST command and select option 17.
7. On each queue manager, save the latest media recovery checkpoint. To do so, enter the following command:
   ```
   RCDMQMIMG OBJ(*ALL) OBJTYPE(*ALL) MQMNAME( QMGRNAME ) DSPJRNDTA(*YES)
   ```

Quiesce IBM MQ on IBM i: IBM i

Stop all queue managers. If necessary force all queue managers to stop, tidy up shared memory and end all jobs in the QMQM subsystem.

About this task

The orderly shutdown of IBM MQ is called *quiescing*. You need to quiesce IBM MQ to upgrade to a newer version.

To quiesce one or more queue managers:

Procedure
1. Sign on to a new interactive IBM i session, ensuring that you are not accessing any IBM MQ objects.
2. Ensure that you have:
   a. *ALLOBJ authority, or object management authority for the QMQM library.
   b. Sufficient authority to use the ENDSBS command.
3. Warn all users that you are going to stop IBM MQ.
4. Use the ENDMQM command to quiesce all queue managers:
   ```
   ENDMQM MQMNAME(*ALL) OPTION(*CNTRLD) ENDCCTJOB(*YES) RCDMQMIMG(*YES)
   TIMEOUT( 15 )
   ```
   Where *15* is a timeout value in seconds.

   If the ENDMQM command has not completed within a reasonable period (at least 10 minutes), use the WRKMQM command. This command identifies the queue managers that are still ending. Then force each one in turn to stop by issuing:
   ```
   ENDMQM MQMNAME( QMGRNAME ) OPTION(*IMMED)
   ```
   Where *QMGRNAME* is the name of the queue manager.

   Complete the tidying up of shared memory by issuing the command:
   ```
   ENDMQM MQMNAME(*ALL) OPTION(*IMMED) ENDCCTJOB(*YES) RCDMQMIMG(*NO)
   TIMEOUT( 15 )
   ```
5. If the command in the previous step does not complete, end the subsystem immediately by issuing:
   ```
   ENDSBS SBS(QMQM) OPTION(*IMMED)
   ```
6. If this command also fails, use the operating system command ENDJOB to end all jobs in the subsystem QMQM, as described in the following steps.

   Note: Do not use ENDJOBABN unless you intend to perform an IPL on the machine before starting IBM MQ. Ending IBM MQ jobs using ENDJOBABN can lead to damaged semaphores, which in turn can prevent your queue manager from starting.
   a. If a QMGR must be shut down manually, the recommended order of ending jobs (ENDJOB) is shown in the list that follows. Wait a few minutes for AMQA* or AMQZ* jobs to tidy up.
      1) RUNMQLSR - TCP listener (multi-threaded)
      2) AMQCLMAA - TCP listener (single-threaded)
      3) AMQRMPPA - Channel process pooling job

4) `RUNMQCHI` - channel initiator
5) `AMQCRSTA` - receiving MCA jobs
6) `RUNMQCHL` - sending MCA jobs
7) `AMQCRS6B` - LU62 receiver channel
8) `AMQPCSEA` - command server
9) `RUNMQTRM` - Application trigger monitor
10) `RUNMQDLQ` - Dead letter queue handler
11) `AMQFCXBA` - IBM Integration Bus Worker Job
12) `AMQFQPUB` - Queued Publish/Subscribe Daemon
13) `RUNMQBRK` - IBM Integration Bus Control Job
14) `AMQZMUC0` ('0' is a zero) - Utility Manager
15) `AMQZMUF0` ('0' is a zero) - Utility Manager
16) `AMQZMUR0` ('0' is a zero) - Utility Manager
17) `AMQZMGR0` ('0' is a zero) - Process Controller
18) `AMQRRMFA` - cluster repository manager
19) `AMQZDMAA` - deferred message manager
20) `AMQALMPX` - Log Manager
21) `AMQZFUMA` - object authority manager
22) `AMQZLSA0` ('0' is a zero) - LQM agents
23) `AMQZLAA0` ('0' is a zero) - LQM agents
24) `AMQZXMA0` ('0' is a zero) - Execution Controller

b. Issue the following command:

```
ENDMQM MQMNAME( QMGRNAME ) OPTION(*IMMED)
```

c. Issue the following command:

```
ENDMQM MQMNAME(*ALL) OPTION(*CNTRLD) ENDCCTJOB(*YES) RCDMQMIMG(*NO)
TIMEOUT( 05 )
```

Where *05* is a timeout value in seconds.

d. Manually clean up shared memory. Issue the following command:

```
EDTF '/QIBM/UserData/mqm/qmgrs'
```

then:

1) Take option 5 for **&SYSTEM** and check that the following directories are empty: `isem`, `esem`, `msem`, `ssem`, and `shmem`.
2) Take option 5 for **QMGRNAME** and check that the following directories are empty:- `isem`, `esem`, `msem`, `ssem`, and `shmem`.
3) Take option 5 for **&ipcc** in the QMGRNAME directory and check that the following directories are empty:- `isem`, `esem`, `msem`, `ssem`, and `shmem`.
4) Take option 5 for **&qmpersist** in the QMGRNAME directory and check that the following directories are empty:- `isem`, `esem`, `msem`, `ssem`, and `shmem`.
5) Take option 5 for **&app** and check that the following directories are empty: `isem`, `esem`, `msem`, `ssem`, and `shmem`.

Save IBM MQ data on IBM i: IBM i

Save IBM MQ data after removing unwanted FDC, trace, and JOB files.

Before you begin

You need to have completed the tasks to remove unwanted and indoubt messages and quiesced IBM MQ.

About this task

Procedure

1. Create a save file for every queue manager library on your system. To do so, issue the command:
   ```
   CRTSAVF FILE(QGPL/ queue_manager_library )
   ```
   where the *queue_manager_library* name consists of the name of the queue manager preceded by `QM`.
2. Save your queue manager libraries into the save files. To do so, issue the commands:
   ```
   SAVLIB LIB( queue_manager_library ) DEV(*SAVF)
   SAVF(QGPL/ queue_manager_library )
   ```
3. Remove all unwanted FDC data from directory:
   ```
   QIBM/UserData/mqm/errors
   ```
4. Remove old FDC files with the command:
   ```
   RMVLNK OBJLNK('/QIBM/UserData/mqm/errors/*.FDC')
   ```
   This command cleans up all files with an extension of 'FDC' in the IFS.
5. Remove old JOB files with the command:
   ```
   RMVLNK OBJLNK('/QIBM/UserData/mqm/errors/*.JOB')
   ```
   This command cleans up all files with an extension of 'JOB' in the IFS.
6. Remove all unwanted trace data from directory, or remove the whole directory:
   ```
   QIBM/UserData/mqm/trace
   ```
7. Remove all trace files with the command:
   ```
   RMVLNK OBJLNK('/qibm/userdata/mqm/trace/*')
   ```
8. Create a save file for IBM MQ IFS data. To do so, issue the command:
   ```
   CRTSAVF FILE(QGPL/QMUSERDATA)
   ```
9. Save your IBM MQ IFS data, using the command:
   ```
   SAV DEV('/QSYS.LIB/QGPL.LIB/QMUSERDATA.FILE') OBJ('/QIBM/UserData/mqm')
   ```
10. If you are going to run IBM MQ on a new machine, transfer the save files to the new machine.

Install IBM MQ server on IBM i: IBM i

Install the IBM MQ server in its primary language.

Before you begin

You have completed planning the installation, obtained the installation disks, and set the system values; see Setting system values.

About this task

Install the IBM MQ server and force object conversion. Object conversion migrates objects from the older to the newer version. By performing it now, rather than when an object is first used, you avoid slowing down the first use of the upgraded product.

After following the optional step to pre-agree the license, the **RSTLICPGM** command runs without requiring any interactive input. Otherwise the license agreement is displayed for you to accept. See License requirements.

Procedure

1. Sign on to the system with a user profile that has `*ALLOBJ` special authority, for example `QSECOFR`.
2. Optionally pre-agree the license terms and conditions by running the command,
   ```
   CALL PGM (QSYS/QLPACAGR) PARM ('5724H72' 'V8R0M0' '0000' 0)
   ```
   Where the parameters of **PARM** are,

   5724H72
   The product identifier for IBM i.

   V8R0M0
   The version, release, and modification level.

   0000
   The option number for the *BASE IBM MQ product option.

   0 Unused error structure.
3. Install IBM MQ for IBM i, base product, and primary language.
   ```
   RSTLICPGM LICPGM (5724H72) DEV (installation device) OPTION (*BASE) OUTPUT (*PRINT)
   ```
   where the parameters of RSTLICPGM are,

   LICPGM (5724H72)
   The product identifier for IBM i.

   DEV (*installation device*)
   The device from which the product is to be loaded, typically an optical drive, for example, `OPT01`.

   OPTION (*BASE)
   Install the base IBM MQ for IBM i product.

   Unspecified parameters
   Unspecified parameters such as **RSTOBJ** (`*ALL`), revert to defaults. The command installs both IBM MQ and the language files for the primary language of your system. For installing additional languages see Installing translated versions.

What to do next

Install any Progam Temporary Fixes (PTF) that have been issued.

Install samples on IBM i: ▶ IBM i

Install the IBM MQ samples

Before you begin

If you have not already done so, sign on to the system with a user profile that has `*ALLOBJ` special authority, for example `QSECOFR`.

About this task

Install the samples.

After following the optional step to pre-agree the license, the **RSTLICPGM** command runs without requiring any interactive input. Otherwise the license agreement is displayed for you to accept. See License requirements.

Procedure

1. Optionally pre-agree the license terms and conditions by running the command,

   ```
    CALL PGM (QSYS/QLPACAGR) PARM ('5724H72' 'V8R0M0' '0001' 0)
   ```

   Where the parameters of **PARM** are,

   5724H72
   The product identifier for IBM i.

   V8R0M0
   The version, release, and modification level.

   0001
   The option number for the samples.

   0 Unused error structure.

2. Install the samples using the command:

   ```
   RSTLICPGM LICPGM (5724H72) DEV (installation device) OPTION (1) OUTPUT (*PRINT)
   ```

   Where the parameters of RSTLICPGM are,

   LICPGM (5724H72)
   The product identifier for IBM i.

   DEV (*installation device*)
   The device from which the product is to be loaded, typically an optical drive, for example, `OPT01`.

   OPTION (1)
   Install the samples for IBM i.

   OUTPUT (*PRINT
   The output is printed with the spooled output of the job.

Install translated versions on IBM i: IBM i

Install translated versions of IBM MQ from a choice of national-languages.

About this task

The following language versions are available for IBM i:

Table 87. National-language versions of IBM MQ for IBM i

Language ID	Language
2909	Belgian English
2966	Belgian French MNCS (Multi-National Character Set)
2981	Canadian French MNCS
2975	Czech
2950	English uppercase
2924	English uppercase and lowercase
2984	English US DBCS
2938	English US uppercase DBCS
2928	French
2940	French MNCS
2929	German
2939	German MNCS
2976	Hungarian
2932	Italian
2942	Italian MNCS
2962	Japanese
2986	Korean
2978	Polish
2979	Russian
2989	Simplified Chinese
2931	Spanish

IBM MQ for IBM i is installed in the language that is the primary language on your system.

You can install additional versions of the product in any of the languages shown in Table 87. To do so:

Procedure

1. Sign on to the system with a user profile that has *ALLOBJ special authority
2. Issue the following command specifying the appropriate language ID:
   ```
   RSTLICPGM LICPGM(5724H72) DEV( installation device ) RSTOBJ(*LNG) LNG( language ID )
   ```

   This installs the commands, message file, and panel groups into the relevant QSYS library for the language. For example, library QSYS2928 is used for French. If this QSYS29nn library does not exist, it is created by the RSTLICPGM command.

Results

Note:

1. To run the Japanese language version of IBM MQ for IBM i, the CCSID of the job must be 939 (5035) rather than 930 (5026) because IBM MQ uses lowercase English characters.
2. If you are installing IBM MQ for IBM i onto a machine for which the primary language is not on the CD, the installation program prompts you to load a CD containing the product in that language. If, however, you have only one product CD, this means that the IBM MQ product has not been translated into your language. To get around this issue, proceed as follows:
   - Install the product in one of the supplied languages, and then add the corresponding QSYS29nn library into the *system library* list (for example using command CHGSYSLIBL). At the same time, check that there are no IBM MQ *CMD, *MENU, or *MSGF objects in libraries higher up the library list. If some exist, then either delete these objects (because they refer to an earlier version of IBM MQ) or reorder the System Library list (because the product has been installed in more than one of the supplied languages).

Verify the installation on IBM i: IBM i

How to check that your installation has been successful.

Procedure

1. To ensure that the product has loaded correctly, issue the Display Software Resources (DSPSFWRSC) command and check that the licensed program `5724H72` is listed. If you have installed the base and the optional samples, you see:

```
Resource
ID    Option Feature Description
5724H72  *BASE  5050 IBM MQ for IBM i
5724H72  *BASE  2924 IBM MQ for IBM i
5724H72  1    5050 IBM MQ for IBM i - Samples
```

2. Press F11, while viewing the Display Software Resources screen, and you see the library and version number of the products installed:

```
Resource         Feature
ID    Option Feature  Type  Library  Release
5724H72  *BASE  5050   *CODE  QMQM   V9R0M0
5724H72  *BASE  2924   *LNG   QMQM   V9R0M0
5724H72  1    5050   *CODE  QMQMSAMP V9R0M0
```

3. If you have installed additional language versions, you also see entries for these versions. For example, if you have installed the French version, for which the language ID is 2928, you see:
   a.
   ```
   Resource
   ID    Option Feature Description
   5724H72  *BASE  2928 IBM MQ for IBM i
   ```
   b. and when you press F11:
   ```
   Resource          Feature
   ID    Option Feature  Type  Library   Release
   5724H72  *BASE  2928  *LNG  QSYS2928  V9R0M0
   ```

4. Use the command DSPMQMVER to check exactly what version you have installed. For example, for V9R0M0, it reports:

```
Version: 9.0.0.0
```

Verify the upgrade on IBM i: IBM i

After you have verified the installation, start the IBM MQ subsystem, check the queue managers, and take a fresh media recovery checkpoint.

About this task

To verify that you have migrated to the latest version IBM MQ for IBM i, successfully:

Procedure

1. Make QMQMADM either the primary or a secondary group profile for your user profile. To do so, issue one of the following commands:
   ```
   CHGUSRPRF USRPRF( YOUR PROFILE ) GRPPRF(QMQMADM)
   CHGUSRPRF USRPRF( YOUR PROFILE ) SUPGRPPRF(QMQMADM)
   ```
2. Start the IBM MQ subsystem with the command:
   ```
   STRSBS SBSD(QMQM/QMQM)
   ```
   (If it is already running, you get error message `CPF1010` which you can safely ignore).
3. Check that your queue managers are accessible by issuing the command:
   ```
   WRKMQM
   ```
   Use option 14 against each queue manager to start it.

   Use option 5 against each queue manager to check its attributes.
4. You can use the other options to check your queue manager objects. For example, check your queues using option 18, check your channels using option 20, and so on.
5. Take a fresh media recovery checkpoint, using the following command:
   ```
   RCDMQMIMG OBJ(*ALL) OBJTYPE(*ALL) MQMNAME( QMGRNAME ) DSPJRNDTA(*YES)
   ```
   Where *QMGRNAME* is the name of the queue manager.

Restore queue managers after upgrading IBM MQ on IBM i: IBM i

Complete the side-by-side upgrade by restored the saved queue managers onto the server that you have upgraded.

Before you begin

Note: Carry out this task only if you are performing a side-by-side upgrade.

Ensure that you have saved your queue manager data, see "End IBM MQ activity on IBM i" on page 752, and installed and verified the upgrade.

About this task

Transfer the queue manager data, and journal receivers, onto the server that has been upgraded.

Procedure

1. Restore the queue manager libraries for every queue manager, using the command:
   ```
   RSTLIB SAVLIB( queue_manager_library ) DEV(*SAVF) (*PRINT)
   SAVF(QGPL/ queue_manager_library )
   ```

where the *`queue_manager_library`* name consists of the name of the queue manager preceded by `QM`.

2. Restore the IBM MQ IFS data, using the command:
   ```
   RST DEV('/QSYS.LIB/QGPL.LIB/QMUSERDATA.FILE') OBJ('/QIBM/UserData/mqm') (*PRINT)
   ```
3. To associate the journal receivers, issue the command `WRKJRN` on the journal `AMQAJRN` in each queue manager library, by pressing *PF4* and selecting option 9.
4. If you want to set up your work management environment, job descriptions, and pools, see the Administering IBMi for guidance. Otherwise, use the default setup.

After Upgrading on IBM MQ for IBM i: IBM i

Tasks to perform after you have upgraded IBM MQ for IBM i.

About this task

Satisfy yourself the upgrade has completed successfully.

Procedure

Delete the saved data in the save files in QGPL. This data was saved in "Save IBM MQ data on IBM i" on page 755.

Post installation tasks for IBM i: IBM i

Tasks to perform after you have installed IBM MQ for IBM i, and before using it.

About this task

When you have correctly installed IBM MQ for IBM i on your system:

Procedure

1. For the latest product information for IBM i, see System requirements for IBM MQ .
2. To install and apply all fix packs, see "Applying maintenance level updates on IBM i" on page 613.
3. Where you have more than one system and a mixture of releases of OS/400 or IBM i, and IBM MQ, you must take care when compiling CL programs. You must compile CL programs either on the system they are to run on, or on one with an identical combination of releases of OS/400 or IBM i, and IBM MQ. When you install later versions of IBM MQ, delete all IBM MQ commands from previous releases in any QSYSVvRrMm libraries using the QSYS/DLTCMD command.
4. If you have not installed IBM MQ on your system before, you must add user profiles to the `QMQMADM` group profile. Make all user profiles that are to be used for creating and administering queue managers members of the `QMQMADM` group profile, using the command `CHGUSRPRF`.
   a. Start the IBM MQ subsystem, by issuing the command:
      ```
      STRSBS SBSD(QMQM/QMQM)
      ```
      Note: The subsystem must be started after each IPL of the system, so you might choose to start it as part of your system startup process.
5. Create the system-default objects. The system-default objects are created automatically when you issue the CRTMQM command to create a queue manager. For example: `CRTMQM MQMNAME(QMGRNAME) ASP(*SYSTEM)`. You can refresh them using the STRMQM command (Warning: this command will replace any existing default objects). For example: `STRMQM MQMNAME(QMGRNAME) RDEFSYS(*YES)`. Refer to the onscreen help for information about using this command.

   Note: on the command `STRMQM MQMNAME(QMGRNAME) RDEFSYS(*YES)`:

- The command does not re-create the objects, it performs a CRTxxxx REPLACE(*YES) for all of the SYSTEM.* objects.
- This means that it refreshes the parameters on the objects back to their defaults. So if, for example, on the SYSTEM.DEFAULT.LOCAL.QUEUE object, TRGENBL had previously been changed to *YES, then, when the command is run, it is changed back to TRGENBL(*NO).
- If any messages exist on a queue, they are left intact, because the queues are not physically deleted.
- The contents of the SYSTEM.AUTH.DATA.QUEUE are untouched when this command is run.
- So, if the contents of this (or any other significant queue) become corrupt, it must be physically deleted and re-created either from scratch, or from a backup.

Results

You are now ready to start using IBM MQ for IBM i.

Note: When you install IBM MQ for IBM i, two user profiles are created:
- QMQM
- QMQMADM

These two objects are central to the correct running of IBM MQ for IBM i. Do not alter or delete them. If you do, IBM cannot guarantee correct behavior of your product.

If you uninstall IBM MQ and data, these profiles are deleted. If you uninstall IBM MQ only, these profiles are retained.

Migrating a queue manager to a later version on IBM i - alternative method

IBM i

An alternative method of migrating a queue manager from an earlier version to a later version

Before you begin

1. Review the IBM MQ system requirements for the later version of the product; see System Requirements for IBM MQ
2. Review any other installed SupportPacs for their applicability to the later version of IBM MQ.

About this task

There are various parts to this form of migration:
1. As part of upgrading the IBM MQ product, carry out the following tasks:
   a. “Preparing to install IBM MQ on IBM i” on page 763
   b. “Install IBM MQ server on IBM i” on page 763
2. Following the IBM MQ product upgrade, carry out the following task:
   a. “Post installation tasks” on page 764

Preparing to install IBM MQ on IBM i: IBM i

Carry out the following tasks to prepare your system for an upgrade.

Procedure

1. Stop the IBM MQ queue managers by issuing the following command:

   ```
   ENDMQM MQMNAME(*ALL) OPTION(*IMMED) ENDCCTJOB(*YES) RCDMQMIMG(*YES)
   TIMEOUT(30)
   ```

   Ensure that the user profile issuing this command has `*ALLOBJ` authority.
2. Create a save file for every queue manager library on your system. To do so, issue the command:

   ```
   CRTSAVF FILE(QGPL/ queue_manager_library )
   ```

   where the *`queue_manager_library`* name consists of the name of the queue manager preceded by `QM`.
3. Save your queue manager libraries into the save files. To do so, issue the commands:

   ```
   SAVLIB LIB( queue_manager_library ) DEV(*SAVF)
   SAVF(QGPL/ queue_manager_library )
   ```
4. Create a save file for IBM MQ IFS data. To do so, issue the command:

   ```
   CRTSAVF FILE(QGPL/QMUSERDATA)
   ```
5. Save your IBM MQ IFS data, using the command:

   ```
   SAV DEV('/QSYS.LIB/QGPL.LIB/QMUSERDATA.FILE') OBJ('/QIBM/UserData/mqm')
   ```
6. If you are going to run IBM MQ on a new machine, transfer the save files to the new machine.
7. Issue the following command before you upgrade your IBM MQ product, only if the upgrade is required on the same machine.
   a. `DLTMQM` *`QMgrName`*
   b. `ENDSBS SBS(QMQM) OPTION(*IMMED)`
   c. `WRKOBJLCK OBJ(QMQM) OBJTYPE(*LIB)`

   Relinquish any locks on the system.

Install IBM MQ server on IBM i: IBM i

Install the IBM MQ server in its primary language and force object conversion.

Before you begin

In either of the following cases, ensure that you have completed the planning and set the system values; see Setting system values

- If you have obtained the product through IBM passport advantage, follow the instructions in the `EGA.README.txt` file.
- If you have obtained the product on disk, follow the instructions within this topic.

About this task

Install the IBM MQ server and force object conversion. Object conversion migrates objects from the older to the newer version. By performing it now, rather than when an object is first used, you avoid slowing down the first use of the upgraded product.

After following the optional step to pre-agree the license, the **`RSTLICPGM`** command runs without requiring any interactive input. Otherwise the license agreement is displayed for you to accept. See License requirements.

Procedure

1. Sign on to the system with a user profile that has `*ALLOBJ` special authority, for example `QSECOFR`.
2. Optionally pre-agree the license terms and conditions by running the command,

   ```
   CALL PGM (QSYS/QLPACAGR) PARM ('5724H72' 'V8R0M0' '0000' 0)
   ```

   Where the parameters of **PARM** are,

   5724H72
   The product identifier for IBM i.

   V8R0M0
   The version, release, and modification level.

   0000
   The option number for the *BASE IBM MQ product option.

   0 Unused error structure.
3. Install IBM MQ for IBM i, base product, and primary language.

   ```
   RSTLICPGM LICPGM (5724H72) DEV (installation device) OPTION (*BASE) OUTPUT (*PRINT)
   ```

   where the parameters of RSTLICPGM are,

   LICPGM (5724H72)
   The product identifier for IBM i.

   DEV (*installation device*)
   The device from which the product is to be loaded, typically an optical drive, for example, `OPT01`.

   OPTION (*BASE)
   Install the base IBM MQ for IBM i product.

   Unspecified parameters
   Unspecified parameters such as **RSTOBJ** (`*ALL`), revert to defaults. The command installs both IBM MQ and the language files for the primary language of your system. For installing additional languages see Installing translated versions.

What to do next

Install any Progam Temporary Fixes (PTF) that have been issued.

To install the IBM MQ samples, see: "Install samples on IBM i" on page 757.

Post installation tasks:

Actions required after upgrading IBM MQ.

About this task

Install the samples.

Carry out these steps after installing the product.

Procedure

1. Issue the following commands:
   a. `STRSBS SBSD(QMQM/QMQM)`
   b. `CRTMQM MQMNAME(`*`QMgrName`*`) DFTQMGR(*YES)` You receive the message " IBM MQ queue manager created."

   c. `STRMQM MQMNAME(`*`QMgrName`*`)` You receive the message " IBM MQ queue manager '*QMgrName*' started."
2. Issue the following command:
   ```
   STRMQMMQSC SRCMBR(QMgrName) SRCFILE(*CURLIB/QMQSC) OPTION(*RUN)
   MQMNAME(QMgrName)
   ```
3. Reapply IBM MQ Authorities by issuing the command: `CALL PGM(*CURLIB/`*`QMgrName`*`)`
   a. You must compile the CLP as follows:
      ```
      CRTCLPGM PGM(*CURLIB/QMgrName) SRCFILE(*CURLIB/QMAUT) SRCMBR(*PGM)
      ```

Upgrading an entire IBM MQ system on IBM i

IBM i

How to upgrade an IBM MQ system on IBM i

Before you begin

Ensure that you have backed up your entire system.

About this task

To upgrade an IBM MQ system on IBM i you carry out a slip installation.

See “Installation methods on IBM i” on page 752 for further information.

Related tasks:

“Migrating a queue manager on Windows” on page 691
The procedures for migrating a queue manager to a later version of the product, and for restoring a queue manager to an earlier version of the product are detailed in this section.

Migrating an IBM MQ MQI client to the latest version on IBM i

IBM i

Before migrating an IBM MQ MQI client, create a migration plan. Stop all IBM MQ activity on the client workstation. Upgrade the IBM MQ MQI client installation. Make any essential configuration and application changes.

Before you begin

1. Create a migration plan. Use the planning task, “Planning to migrate to the latest version on IBM i” on page 749, as a guide.

Procedure

1. Review the IBM MQ system requirements for the later version of the product. See System Requirements for IBM MQ.
2. Review all the changes in IBM MQ that affect you. For more information, see “Changes that affect migration” on page 626.
3. End all IBM MQ activity on the workstation.
4. Upgrade the client. To upgrade an IBM MQ MQI client for IBM i installation on a workstation; see Client installation procedure on IBM i.

What to do next

Complete the tasks in your migration plan, such as verifying IBM MQ MQI client applications work correctly with the latest version.

Related concepts:

"IBM MQ MQI client migration" on page 650
IBM MQ MQI client migration is the process of converting IBM MQ MQI client configurations, and client and server channels from one version to another. Client migration can take place after upgrading the IBM MQ MQI client, and is reversible.

Related tasks:

UNIX Linux "Migrating an IBM MQ MQI client on UNIX and Linux" on page 736
Before migrating an IBM MQ MQI client, create a migration plan. Stop all IBM MQ activity on the client workstation. Upgrade the IBM MQ MQI client installation. Make any essential configuration and application changes.

Windows "Migrating an IBM MQ MQI client on Windows" on page 707
Before migrating an IBM MQ MQI client, create a migration plan. Stop all IBM MQ activity on the client workstation. Upgrade the IBM MQ MQI client installation. Make any essential configuration and application changes.

Related information:

Client installation procedure on IBM i

Installing IBM MQ MQI clients on the same machine as the server

Migrating from a single instance to a multi-instance queue manager on IBM i

IBM i

To migrate a single instance queue manager to a multi-instance queue manager, on IBM i, 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. Some environments have been tested with multi-instance queue managers, and are known to work. IBM i has been tested with multi-instance queue managers and is known to work See Testing and support statement for IBM MQ multi-instance queue managers for the latest list of tested environments. The support statement has detailed version and prerequisite information for each environment it lists. 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 and support statement for IBM MQ multi-instance queue managers. 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 **amqmfsck** 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 **amqmfsck** 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 IBM i, `QMQM`, `QMQMADM`, and any other user profiles that are granted access to the share must have the same passwords on all the servers
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 (Version 7.0.1 onwards) in Example directory configurations on UNIX.

   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` with the following properties:

   On IBM i, follow the instructions to create a network share using NetServer.
4. Copy the queue manager data and the logs to the share.

   You can choose to copy files manually, by following the procedure to back up the queue manager. elect one of these methods:
   - Follow the instructions in Backups of IBM MQ for IBM i data, copying the queue manager data to the share. You must use this method if the **DataPath** configuration attribute is specified for this queue manager.
   - Stop the queue manager, and then type the command,
     ```
     hamvmqm /m /dd share\data /dd share\logs
     ```
     Where *share* is to be the location of the data and logs that you created in step 3.
5. Update the queue manager configuration information stored on the current queue manager server.

   If you moved the queue manager data and logs by running the **hamvmqm** command, the command has already modified the configuration information correctly for you.

   If you moved the queue manager data and logs manually, you must complete the following steps.
   - On IBM i,
     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 `/QIBM/UserData/mqm` directory on IBM i:

      ```
      DataPath= share/data/QMgrName
      ```

   Where, *QMgrName* is the `Directory` name in the `QueueManager:` stanza in the `mqs.ini` file on IBM i. *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 in Version 6.0.

      ```
      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 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:

      ```
      echo DISPLAY CHANNEL(ENGLAND) CONNAME | runmqsc QM1
      5724-H72 (C) Copyright IBM Corp. 1994, 2009.  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 information:

`amqmfsck` (file system check)

Application recovery

Automatic client reconnection

Backing up queue manager data

Channel and client reconnection

Multi-instance queue managers

IBM i Multi-instance queue managers on IBM i

Shared file system

Testing a shared file system for compatibility with IBM MQ Multi-instance Queue Managers

Testing and support statement for IBM MQ multi-instance queue managers

IBM i The IBM MQ configuration file mqs.ini - IBM i

Verifying shared file system locking

Reverting to a single-instance queue manager on IBM i

IBM i

Revert a multi-instance queue manager to a single instance queue manager, on IBM i, 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 MQMNAME (QMgrName) *WAIT
   ```
2. Stop the active queue manager instance.

   On the server running the active instance:
   ```
   ENDMQM MQMNAME (QMgrName) *WAIT
   ```
3. Restart the queue manager, preventing standbys.

   On the server that is going to run the queue manager:

```
STRMQM MQMNAME (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 "Migrating from a single instance to a multi-instance queue manager on IBM i" on page 766 to create a multi-instance queue manager.

Migrating IBM MQ on z/OS

z/OS

Migration tasks associated with z/OS are grouped in this section.

Related concepts:

"Migration concepts and methods" on page 645
An overview of the various concepts and methods for migrating from one release of the product to another.

Related tasks:

IBM i "Migrating IBM MQ on IBM i" on page 749
IBM MQ migration tasks associated with IBM i are grouped in this section.

UNIX Linux "Migrating IBM MQ on UNIX and Linux" on page 720
Migration tasks associated with UNIX and Linux platforms are grouped in this section.

Windows "Migrating IBM MQ on Windows" on page 684
IBM MQ migration tasks associated with Windows platforms are grouped in this section.

Related reference:

"Changes that affect migration" on page 626
Changes to the product might affect the migration of a queue manager from an earlier release to the current release of IBM MQ, or affect existing applications or configurations. Review these changes before upgrading queue managers to the latest product version and decide whether you must plan to make changes to existing applications, scripts, and procedures before starting to migrate your systems.

Migrating IBM MQ for z/OS - order of tasks

z/OS

Perform these instructions, in the order shown, to migrate a single IBM MQ for z/OS queue manager.

About this task

The tables within this topic show the tasks required in each part of the process to migrate IBM MQ for z/OS, and the order in which these tasks must be done.

Notes:

- You must perform the tasks in the following order:
  1. Before migration
  2. Migrating to the next release
  3. Post-migration tasks

and the order of tasks within each table.

Table 88. Before migration

Task	For your own use
1. Make your configuration ready for migration	
2. Install the new version code	
3. Perform backup	
4. Restart IBM MQ systems	
5. Review pageset 0 usage	
6. Migrate Db2 tables - for each QSG	
7. Add new CF structure definition - for each QSG	
8. Server application migration	
9. Preparing to migrate Advanced Message Security	

Table 89. Migrating to the next release

Task	For your own use
10. Stop or disconnect all your applications	
11. Stop the queue manager	
12. Update STEPLIB for MSTR and channel initiator	
13. Update the initialization input data sets	
CD 14. Update the target version system parameter module	
15. Migrating Advanced Message Security	
16. Review the security control of your system	
17. Start the queue manager	
18. Optionally, revert the queue manager to a previous version	

Table 90. Post migration tasks

Task	For your own use
19. Check the changes in behavior	
20. Modify the backup jobs	
21. Post migration tasks for Advanced Message Security	
22. Perform a full regression test	
23. Update the ZPARM module, if not already done	
24. Set OPMODE to NEWFUNC	
25. Exploit the new function	
26. Consider client application migration	

Planning to migrate to the latest release on z/OS: z/OS

Create a migration plan for IBM MQ for z/OS to upgrade to the latest release.

About this task

The goal of this task is for you to create your own plan to migrate your queue managers to the latest release. Incorporate the task to migrate a queue manager, "Migrating IBM MQ for z/OS - order of tasks" on page 770, into your plan.

Migration plan overview for your enterprise

Migration phase	Required tasks
Phase I, before migration. See "Preparing to migrate a single IBM MQ for z/OS queue manager" on page 791 for further information.	Preparing each queue manager in your enterprise for migration.
Phase II, migrate each single queue manager in the order listed. See "Migrating a single IBM MQ z/OS queue manager to the next release of the product" on page 796 for further information.	Carry out this process for each queue manager in the order shown: 1. z/OS queue managers in a queue-sharing group (QSG). 2. z/OS queue managers not in a queue-sharing group (if any). 3. Queue managers in a full repository on platforms other than z/OS. See "Migrating a queue manager on Windows" on page 691 for further information. 4. z/OS partial repository queue managers in a QSG. 5. z/OS partial repository queue managers not in a QSG . 6. z/OS queue managers not in a queue-sharing group or cluster. 7. Partial repository queue managers on platforms other than z/OS. 8. Queue managers not in a cluster on platforms other than z/OS.
Phase III, post migration. See "Post migration tasks" on page 803 for further information.	Carry out a full regression test, and then explore the new function available to you. Optionally, at any time in the process, upgrade your client libraries if necessary, recompile your clients using the additional features provided by the new version, and deploy the clients.

Procedure

1. Review the IBM MQ system requirements for the latest version.

   See System requirements for IBM MQ.
2. Review all the changes in the product that affect you. For further information, see:
   - LTS What's new and changed in IBM MQ Version 9.0
   - CD What's new and changed in Version 9.0.x Continuous Delivery
3. Review performance changes. See IBM MQ Family - Performance Reports.
4. Review the backward and coexistence (or migration and toleration) PTFs for your current version of the product. See IBM MQ Support, Migration PTFs.

These PTFs must be applied to your current version of the product to enable you to revert your queue managers to the current version, after the queue managers have been started at the target version.

Note, that you can have different versions of queue managers coexisting in the same queue-sharing group.

If you are unsure which migration PTFs you require, run the following command SMP/E:

CD

```
REPORT MISSINGFIX ZONES(mqtgtzone) FIXCAT(IBM.Coexistence.MQ.V9R0M0)
```

See FIXCAT and IBM MQ Migration Installation for further information.

Attention: If a PTF requires a rebind of Db2 plans, the PTF is shipped with ++HOLD(ACTION), indicating the need for this process. In such a case, see Migrating Db2 tables to bind the plans before starting migration.

Other FIXCAT categories are listed in IBM Fix Category Values and Descriptions.

5. Plan to install the latest version early code, and activate for all queue managers on the LPAR. See Installing early code for more information.

   Note that:

      Before migration, all systems that are running queue managers that you plan to migrate to the latest version must have the early code for that version installed and running. Queue managers in queue-sharing groups that contain queue managers that are to be migrated, must also be running the early code.

      A queue manager must use the early code from the same release level, or a later release level.

   If you need to rebind the plans or packages, see Step 1 on page 806 in Migrating queue-sharing groups from a previous release for further information.

6. Consider using aliases for the IBM MQ libraries.

   For example, use the IDCAMS utility with the DEFINE command:

   ```
   DEFINE ALIAS(NAME(MQM.SCSQANLE)RELATE(MQM.V710.SCSQANLE))
   ```

   You can use `MQM.SCSQANLE`, where applicable, in your STEPLIB, and it resolves to the actual data set.

   When you migrate to a new release, change the alias definition, rather than changing all the places in your JCL where the library is referenced.

   This process has the most benefit for your server application programs, because you can get all of the programs to refer to the new libraries at the same time.

7. Plan the sequence and timing of queue manager upgrades.
   - You must install the backward and coexistence (or migration and toleration) PTF to bring the previous version queue managers up to the latest maintenance level for that version.
   - You must install the PTF on all members of a queue-sharing group, before you migrate any queue managers to the latest version. You can install the PTF one member at a time, leaving the other members running.
   - If the queue manager is a member of a queue manager cluster, you must consider the order of migration of queue managers in the cluster; see “Migrating a queue manager cluster” on page 808.
   - Check that any products that require the previous version of the product also support the new version.

8. Plan to update any manual or automated procedures you have written with changes to messages and codes.

9. Plan to update applications that might be affected by changes. Update the IBM MQ library in the application STEPLIB concatenations to the latest version.

   Consider whether the application must be able to run on both the previous version and the latest version. You might be able to change the application to be compatible with both code levels. If you cannot, you can query the queue manager command level, and make the code conditional on the command level. Call MQINQ setting the `MQIA_COMMAND_LEVEL` selector.

10. LTS If you are migrating to a Long Term Support (LTS) release, decide on what regression tests to perform before enabling the new function in the latest version. The **OPMODE** parameter controls a staged migration from the previous version to the latest version.

    LTS Do not change **OPMODE** initially, when migrating to an LTS release, to make sure that you can go back to using an earlier version of the product, and that all the functions that were available before migration will still be available after migration.

    If you are migrating from IBM WebSphere MQ Version 7.1 to IBM MQ Version 9.0, once you are satisfied with the stability of the latest version, you can start to use the new functions. To use the new functions, you must set **OPMODE** to `(NEWFUNC,900)`.

    There are no new functions in IBM MQ Version 9.0 that are controlled by **OPMODE**, therefore if you are migrating from IBM MQ Version 8.0 to IBM MQ Version 9.0, setting **OPMODE** to `(NEWFUNC,900)` does not enable any new functions.

    CD Backwards migration from a Continuous Delivery release CD release is not possible. If you are migrating to a CD release for the first time, you must set **OPMODE** to `(NEWFUNC,90x)` as part of the migration procedure, where `x` is the modification number.

    Include the procedures and applications you identified in steps 8 on page 773 and 9 on page 773 in your regression tests.
11. Review the tasks to customize z/OS, and the queue manager. Plan how to change the queue manager definitions and started task JCL to migrate your queue managers to the latest versions. The customization steps for migration are described in "Reviewing and modifying queue manager customizations from the previous release on z/OS" on page 775.
12. Review the usage of page set 0. Issue the operator command **cpf**, `/cpf DISPLAY USAGE PSID(0)` to obtain a report on pageset 0 usage.

    The size of queue definitions increased in IBM WebSphere MQ Version 7.1. During migration queue definitions stored on pageset 0 are rewritten if you are migrating from a previous release. The rewrite is carried out as a single transaction when the queue manager is first migrated to IBM WebSphere MQ Version 7.1.

    Ensure that there is enough space available on pageset 0 to create a copy of the queue definitions while migration is taking place. Typically, 60% free space on pageset 0 before migration is sufficient. However, the use of `EXPAND(SYSTEM)` on the pageset definition allows for automatic expansion as required. If there is insufficient space on pageset 0 during migration, the queue manager abends with completion code `X'5C6'` and reason code `X'00C91900'`.
13. Check that you are using a supported level of assembler or compiler. You can write IBM MQ applications using any compiler capable of generating standard OS linkage to the IBM MQ stub routines.

    Some of the data types used by IBM MQ API calls are not supported on some older compilers. You might require a more recent compiler. The following limitations are known:
    a. Assembler copy books contain blank lines, which are not tolerated by assemblers earlier than **HLASM**.
    b. Some older releases of PL/I do not support fixed `bin(63)` type. A macro defines such fields as `char(8)` when an earlier compiler is detected.
    c. Some older releases of COBOL do not support function-pointers, which are used by the MQCB API.
14. Plan any changes to libraries required by your applications and channel exits.
15. Plan to upgrade your IBM MQ MQI client installations to the latest version.
16. Plan to upgrade your client and server applications to use new functions in the latest version.

17. Plan to migrate other vendor software, such as WebSphere Application Server, or CICS to use the latest version. Update the IBM MQ libraries in the `STEPLIB` and `DFHRPL` concatenations of your CICS region JCL and restart CICS.

    Note:
    - CICS

      Update the IBM MQ libraries in the `STEPLIB` and `DFHRPL` concatenations of your CICS region JCL and restart CICS.

      Up to, and including CICS 3.2, the connection between IBM MQ and CICS is provided by IBM MQ. You must change the `SCSQCICS` and `SCSQAUTH` libraries in the `DFHRPL` concatenation provided by IBM MQ.

      After CICS 3.2, the connection between IBM MQ and CICS is provided by CICS libraries. Update the libraries, if you are using CICS Transaction Server for z/OS Version 3.2 or later. Without this change, you are not able to use the most recent IBM MQ features. You must change the `SCSQCICS` library in the `DFHRPL` concatenation provided by IBM MQ, and also the `STEPLIB` concatenation.

      Create separate CICS started procedure JCL. For each CICS region that is connected to an IBM MQ queue manager, ensure that there is a separate CICS started procedure JCL.

      This ensures that the modification of reference to a certain version of IBM MQ libraries in the CICS started procedure JCL only has impact for that single CICS region. In this way you can migrate one queue manager, and only the CICS region or regions connected to it, which makes staged migration possible.

      CICS `STEPLIB` has `thlqual.SCSQAUTH`, and `DFHRPL` has `thlqual.SCSQCICS`, `thlqual.SCSQLOAD`, and `thlqual.SCSQAUTH`. For more information, see Setting up the CICS- IBM MQ adapter.
18. Review any other installed SupportPacs for their applicability to the latest version.

What to do next

Do the task, "Reviewing and modifying queue manager customizations from the previous release on z/OS" to migrate a queue manager. If you must restore a queue manager to the previous version, see "Reverting a queue manager to a previous release on z/OS" on page 801.

When you are confident existing applications are running without migration problems on the latest version, plan to update **`OPMODE`** to `(NEWFUNC,900)` to enable new function, if you migrated from IBM WebSphere MQ Version 7.1 to IBM MQ Version 9.0.0 LTS release.

CICS Transaction Server for z/OS, Version 3 Release 2 product documentation

The CICS-IBM MQ adapter

IBM MQ Support, Migration PTFs

IBM MQ - SupportPacs by Product

Reviewing and modifying queue manager customizations from the previous release on z/OS: z/OS

Review the z/OS and IBM MQ customization steps, and change any customizations before starting any queue managers with the libraries of the latest version.

Before you begin

You must change the JCL and configuration definitions for the queue manager when you migrate to the latest version. Make copies of the JCL libraries, configuration definitions, and the started task JCL. Modify the copied files to customize the queue manager definitions and JCL for the latest version.

1. Copy the JCL libraries containing the configuration definitions for your queue manager. You must change some of the members, for example to reference the libraries of the new version.
2. Copy the started task JCL for the queue manager and the channel initiator. You might change these members.

You can continue to run the queue manager on the previous version until it is ready to switch to the latest version. Preparing for the switch is a long process. Switching from the earlier version to the latest version is a quick process. The switch to the latest version occurs when you restart the queue manager:

- Switch to using your customized copies of the configuration definitions and started task JCL for the queue manager.

  Create data set aliases, such as `MQM.MQP1.SCSLOAD`, and reference them in JCL. Map the aliases to the real data sets, such as `MQM.MQV71.SCSLOAD` or `MQM.MQV80.SCSLOAD`.

  Change the aliases to switch between the two sets of target libraries. With the aliases, you can start applications or the queue manager when moving to a new release of IBM MQ without changing the `STEPLIB` JCL; switch the aliases from referring to the earlier version of the product to refer to the latest version of the product.
- Until you have verified the startup, start the queue manager, channel initiator, and listener separately, checking for errors on the console after each component is started. If the startup runs cleanly, combine the startup of all three components in the production environment.

Tip: Create data set aliases such as `MQM.MQP1.SCSLOAD`, and reference them in JCL. Map the aliases to the real data sets, such as `MQM.MQV71.SCSLOAD` or `MQM.MQV80.SCSLOAD`. Change the aliases to switch between the two sets of target libraries. With the aliases, you can start applications or the queue manager when moving to a new release of IBM MQ without changing the `STEPLIB` JCL.

Tip: You can use the z/OS command `D GrS,system,RES=(*,MQM.V701.SCSQLOAD)` to display which jobs are using the specified data set, and so identify which jobs and JCL need to be changed.

About this task

The steps in this task can be performed by restarting a queue manager once. You must restart the queue manager to apply essential maintenance, and to install early code before proceeding with migration.

The steps are based on the setup procedure for new queue managers; see Customizing your queue managers.

Procedure

Steps 1 to 10 on page 778 are z/OS customization tasks.
Steps 11 on page 778 to 16 on page 779 are IBM MQ customization tasks.

1. Review that all the load libraries that must be APF authorized are authorized.

   See Task 2: APF authorize the IBM MQ load libraries.
2. Update the LPA libraries with the new version of early code libraries.

   See Task 3: Update the z/OS link list and LPA.

   This step is unnecessary if you update the IBM MQ early code in all the LPARs as part of bringing your previous version IBM MQ libraries up to the latest maintenance level; see step 3 on page 777.

   You must activate the early code with an IPL, or restart the queue manager after issuing the following command.

```
REFRESH QMGR TYPE(EARLY)
```

Note: As part of the update you are asked to stop the queue manager. While the queue manager is stopped, you must check the *qmgr*.REFRESH.QMGR security profile is set up, refresh the queue manager and restart it.

3. Make your configuration of the previous version ready for migration.
   a. Apply current maintenance to the current version libraries.

      Refer to the Preventive Service Planning (PSP) bucket for the current version of your product; see PSP Buckets - How to find them on Web.
   b. Apply the latest version migration and toleration [9] PTFs to the previous version code; see IBM MQ Support, Migration PTFs.
   c. If required by any of the PTFs, make Db2 configuration changes and rebind Db2.

      See the appropriate section in Task 9: Select and set up your coupling facility offload storage environment.
   d. Install the early code for the latest version.

      Replace the early code.
   e. Make the latest version early code and earlier version target libraries available on all the LPARs that are running queue managers.

      If the queue manager is a member of a queue-sharing group, update all the systems in the group.
   f. Restart IBM MQ systems.

      The early code is activated by an IPL, or a by issuing the command `REFRESH QMGR TYPE(EARLY)`, and restarting the queue manager.
   g. Verify correct function before proceeding and review any special actions detailed in the PTFs.

   If you require fall back at this stage, use normal maintenance procedures to revert to the code for the previous version before PTF application.
4. Update your procedure to start the queue manager.

   Change the `STEPLIB` for the queue manager to reference the libraries of the new version of the product.

   See Task 6: Create procedures for the IBM MQ queue manager.

   IBM MQ uses z/OS memory objects above the bar for some functions. You must allow the queue manager to access storage above the bar.

   Your installation might have customized the `SMFPRMxx` member of `SYS1.PARMLIB`, or the **`IEFUSI`** exit to provide a default limit for jobs using virtual storage above the 2 GB bar. Check these limits give sufficient memory for a queue manager. A reasonable starting allocation is 2GB. The message CSQY220I displays the amount of virtual storage currently used and available.

   If your installation does not have a default limit for storage above the bar, or if you want to use a different limit for your queue manager, you can provide a queue manager-specific restriction on the amount of virtual storage available above the bar for memory objects by coding a **`MEMLIMIT`** parameter on the JCL of the queue manager stored procedure, `xxxxMSTR`, for example:

```
//PROCSTEP EXEC PGM=CSQYASCP,REGION=0M,MEMLIMIT=2G
```

   `MEMLIMIT` defines memory available above the bar; see Address space storage for more information.

   For specific information on setting `MEMLIMIT`, when using the accounting and statistics changes in IBM MQ Version 8.0, see Channel initiator storage usage.
5. Update your procedures for the channel initiator.

   Change the `STEPLIB` for the channel initiator to reference the libraries of the new version of the product.

   See Task 7: Create procedures for the channel initiator.

9. The “migration and toleration” PTFs are also known as the “backward migration and coexistence” PTFs. They are the same PTFs.

6. Review any automated alerts when queue manager and channel initiator messages, and errors, are detected. New messages might have been added that cause automated alerts, and some messages might have changed.
7. Review C language channel exits

   Ensure your C language channel exits are using the following statement:

```
#pragma environment(function-name)
```

   as defined in the C systems programming environment for system exits, described in the z/OS C/C++ Programming Guide.
8. Update the IBM MQ Db2 configuration.

   If you are using queue-sharing groups, you must update the procedures.

   If all queue managers in the Db2 tables are at IBM WebSphere MQ Version 7.1 or earlier, customize and run the CSQ4570T and CSQ4571T sample JCL in hlq.SCSQPROC.

   Customize and run the `CSQ45BPL` and `CSQ45GEX` samples in `hlq.SCSQPROC`. Tailor these members to your environment, using your Db2 subsystem names and data set names.

   `CSQ45BPL` of `hlq.SCSQPROC` contains the plan names required for the latest version of IBM MQ. `CSQ45GEX` of `hlq.SCSQPROC` contains the authorities required.

   See steps 5 and 6 of Task 9: Select and set up your coupling facility offload storage environment.
9. Review your security controls for queue-sharing groups, the channel initiator, and all queue managers accessing the coupling facility list structures.

   See Task 11: Implement your ESM security controls.

   You must ensure that you have addressed the following points when you migrate your security profiles to the latest version of IBM MQ.

   - The External Security Manager software is at the correct version and level and that all of your prerequisite software is installed.
   - IBM MQ security classes have been updated to include the mixed case classes.
   - Enterprise has migrated to mixed case security; see "z/OS Migrating a queue manager to mixed case security" on page 799.
10. Change `SYS1.PARMLIB` to ensure that any changes you made dynamically remain in effect after an IPL.

    `SYS1.PARMLIB` must reference the initialization input data sets that you customize in step 11.

    See Task 12: Update SYS1.PARMLIB members.
11. Update the initialization input data sets.

    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 Data Definition (DD) names `CSQINP1` and `CSQINP2` defined in the queue manager started task procedure.

    See Task 13: Customize the initialization input data sets.

    The `CSQINP1` and `CSQINP2` initialization input data sets include more samples and the contents of some samples have been moved to other samples. Particular changes to take note of are the commands to define queues to hold publish/subscribe state information. The commands must be in the right order.

    You must review the customization you have made previously to `CSQINP1` and `CSQINP2`, and merge them into the initial definitions provided with the latest version of the product.

    Secure the server-connection channels used by clients; see Securing remote connectivity to the queue manager.

    DEFINE SUB for SYSTEM.DEFAULT.SUB is no longer permitted in the CSQINP2 input data set. DEFINE SUB commands can instead be issued from the CSQINPT input data set. The CSQINPT input data set is processed each time the publish/subscribe engine is started, either during queue manager startup, or when the publish/subscribe engine is started with the **`ALTER QMGR PSMODE(ENABLED)`** command. See Issuing commands to IBM MQ for z/OS for more information on using the CSQINPT input data set.

12. Update your system parameter module.

    Your system parameter module is based on the default module, `CSQZPARM` and `CSQ4ZPRM`.

    See Task 17: Tailor your system parameter module.

13. Update the libraries you added to STEPLIB concatenations to make Batch, TSO, and RRS adapters available to applications.

    Change the `STEPLIB` for the Batch, TSO, and RRS adapters to reference the libraries of the new version of the product.

    See Task 19: Set up Batch, TSO, and RRS adapters.

    Note: You can connect applications that reference the latest version `STEPLIB` to a queue manager that is running on the latest version, or an earlier version. You must not connect applications that reference a `STEPLIB` from an earlier version to a queue manager running on a later version.

14. Update the libraries you added to connect CICS to the queue manager.

    You must update the IBM MQ libraries in the STEPLIB and DFHRPL concatenations of your CICS region JCL and restart CICS. You are then able to use the most recent IBM MQ features.

    Note:

    - CICS

      Update the IBM MQ libraries in the `STEPLIB` and `DFHRPL` concatenations of your CICS region JCL and restart CICS.

      Up to, and including CICS 3.2, the connection between IBM MQ and CICS is provided by IBM MQ. You must change the `SCSQCICS` and `SCSQAUTH` libraries in the `DFHRPL` concatenation provided by IBM MQ.

      After CICS 3.2, the connection between IBM MQ and CICS is provided by CICS libraries. Update the libraries, if you are using CICS Transaction Server for z/OS Version 3.2 or later. Without this change, you are not able to use the most recent IBM MQ features. You must change the `SCSQCICS` library in the `DFHRPL` concatenation provided by IBM MQ, and also the `STEPLIB` concatenation.

      Create separate CICS started procedure JCL. For each CICS region that is connected to an IBM MQ queue manager, ensure that there is a separate CICS started procedure JCL.

      This ensures that the modification of reference to a certain version of IBM MQ libraries in the CICS started procedure JCL only has impact for that single CICS region. In this way you can migrate one queue manager, and only the CICS region or regions connected to it, which makes staged migration possible.

      CICS `STEPLIB` has `thlqual.SCSQAUTH`, and `DFHRPL` has `thlqual.SCSQCICS`, `thlqual.SCSQLOAD`, and `thlqual.SCSQAUTH`. For more information, see Setting up the CICS- IBM MQ adapter.

15. Update the libraries you use to set up the operations and control panels.

    Change the `STEPLIB` for the operations and control panel.

    See Task 20: Set up the operations and control panels.

    Note: You can connect the operations and control panel that references the latest version `STEPLIB` to the queue manager that is running on the latest version, or an earlier version. You must not connect the operations and control panel that references a `STEPLIB` from an earlier version to a queue manager running on a later version.

16. Update system libraries to format IBM MQ dumps using the Interactive Problem Control System (IPCS).

    See Task 21: Include the WebSphere(r) MQ dump formatting member.

What to do next

Return to the parent migration planning task, "Planning to migrate to the latest release on z/OS" on page 772.

Related tasks:

"Migrating a single IBM MQ z/OS queue manager to the next release of the product" on page 796
Carry out the instructions in this topic to migrate a single IBM MQ queue manager on z/OS,

IBM MQ Version 9.0 JCL changes on z/OS: z/OS

Table showing changed members between IBM MQ Version 8.0 hlq.SCSQPROC and the IBM MQ Version 9.0 hlq.SCSQPROC PDS libraries at general availability time.

Notes:

1. Nearly all of the members do show changes between releases, but the majority of those changes are what are termed cosmetic. They are often simply changes to reflect the modification to the release number.
2. Any changes to the samples libraries since general availability are not included.
3. This information can be useful to your administrators when migrating existing queue managers, to make sure that new features are picked up correctly.

Modified members on z/OS

Table 91. IBM MQ for z/OS changed members on z/OS

Member name	Description
CSQ4BSDS	Recommended number of records in BSDS increased to support BSDS version 2. This change was delivered in IBM MQ Version 8.0 in a program temporary fix. Naming convention for active log data sets changed to allow up to 310 active log data sets to be defined.
CSQ4CHIN	Changed to add the following parameters to the JCL JOB statement: • MEMLIMIT=256M, to support the gathering of channel initiator statistics. This change was delivered in IBM MQ Version 8.0 in a program temporary fix. • TIME=NOLIMIT, to indicate that the channel initiator can use the processor for an unlimited amount of time.
CSQ4LREC	Naming convention for active log data sets changed to allow up to 310 active log data sets to be defined.
CSQ4SMFJ	Changed to add a DD statement to describe the output file for formatted data manager page set statistics.
CD CSQ4ZPRM	OPMODE default value updated: • OPMODE=(COMPAT,900), IBM MQ Version 9.0.0 LTS release default value • OPMODE=(NEWFUNC,901), IBM MQ Version 9.0.1 CD release default value ACELIM=0 parameter added. This specifies that the maximum size of the ACE storage pool is unlimited.
CSQ4570T and CSQ4571T	Sample jobs to migrate Db2 resources from IBM WebSphere MQ Version 7.0 and IBM WebSphere MQ Version 7.1 to IBM MQ Version 9.0, updated with IBM MQ Version 9.0 names.
CSQ45BPK and CSQ45BPL	Sample jobs to bind Db2 packages updated to use the IBM MQ Version 9.0 names

Table 91. IBM MQ for z/OS changed members on z/OS (continued)

Member name	Description
CSQ4DTB	Statements added to delete the following Db2 tables used by IBM MQ CSQ.ADMIN_MSGS_BAUX4 CSQ.ADMIN_MSGS_BAUX3 CSQ.ADMIN_MSGS_BAUX2 CSQ.ADMIN_MSGS_BAUX1 CSQ.EXTEND_B_QMGR This change was delivered in IBM MQ Version 8.0 in a program temporary fix
CSQ45GEX	Sample Db2 GRANT job updated to use the IBM MQ Version 9.0 names.

OPMODE on z/OS: z/OS

The availability of new functions and backward migration for IBM MQ for z/OS is controlled by the **OPMODE** parameter in the **CSQ6SYSP** macro. IBM MQ Version 8.0 new functions that are restricted by **OPMODE** are not available at Version 9.0 unless enabled with **OPMODE**. There are no new functions in Version 9.0 that are restricted by **OPMODE**.

Important: CD Backwards migration is not supported for a Continuous Delivery (CD) release. Queue managers running a CD release of IBM MQ must be started with (`OPMODE=(NEWFUNC,90`*x*`)`. For example, a Version 9.0.1 queue manager must be started with `OPMODE=(NEWFUNC,901)`.

Once a queue manager has been started at a CD release of IBM MQ with `OPMODE=(NEWFUNC,901)` or higher, there is no need to change OPMODE every time the queue manager is migrated to a later CD release.

LTS The default setting of OPMODE at Version 9.0.0 is `OPMODE=(COMPAT,900)`, which restricts the set of new functions available. Functions added in IBM MQ Version 8.0 and IBM MQ Version 9.0 that are restricted by **OPMODE** are not available. This makes it possible to revert a queue manager to its earlier release level, if you must do so.

LTS To access all capabilities on a Long Term Support (LTS) release queue manager, change the value of **OPMODE** to `OPMODE=(NEWFUNC,900)`. This prevents the queue manager from being started at an earlier version. Plan to do this in order to exploit all new functions as soon as you are confident that you will not need to revert the queue manager to its earlier release.

LTS

Important: When migrating a queue manager to IBM MQ Version 9.0 LTS, do not recompile the system parameter (ZPARM) module with `OPMODE=(COMPAT,900)`. Doing so stops function working, restricted by **OPMODE** in IBM MQ Version 8.0, that might have been in use before the migration.

This is a change from the migration procedure to previous versions of IBM MQ. Leaving **OPMODE** at its current value retains the ability to revert the queue manager to its earlier release, while also keeping the same set of functions enabled.

If you query the value of **OPMODE** with the command DISPLAY SYSTEM, the result is *compatibility mode, compatibility VRM, function VRM*, where *VRM* is a release level.

The value of compatibility *VRM* is the release level you can revert the queue manager back to. If the queue manager is newly created at Version 9.0.0, then `VRM=900`. If the queue manager was previously run at Version 7.1.0, then `VRM=710`.

The value of *function VRM* is the release level of the new functions that are available. In a queue-sharing group, this depends on the **OPMODE** of the other queue managers in the queue-sharing group as well.

Each queue manager in a queue-sharing group must have **OPMODE** set to **OPMODE**=(NEWFUNC,*vrm*), where *vrm* is 800, 900, or 901, and be restarted in order for any queue manager in the queue-sharing group to use Version 8.0 function.

This means that there are effectively two phases to enabling Version 8.0 new function in a queue-sharing group:

1. The first restart with **OPMODE** set to **OPMODE**=(NEWFUNC,800), or higher, prevents the queue manager from being reverted to run on an earlier release level.

   Queue managers migrated from Version 8.0 to Version 9.0.0 can be restarted with **OPMODE** set to `OPMODE=(NEWFUNC,800)` without being prevented from reverting to run at Version 8.0.
2. The second restart, when all other queue-sharing group members have already been restarted with **OPMODE** set to `OPMODE=(NEWFUNC,800)`, or `OPMODE=(NEWFUNC,900)` on an LTSR queue manager, and therefore cannot revert to run on a release level earlier than Version 8.0, allows Version 8.0 new function to be used.

For example, in a queue-sharing group containing three queue managers, there will be a total of five queue manager restarts required to enable Version 8.0 new function on all queue-sharing group members.

You can reset **OPMODE** to `OPMODE=(COMPAT,900)`, after setting `OPMODE=(NEWFUNC,900)` on an LTS release queue manager, to prevent new functions being used. If you do so, DISPLAY SYSTEM shows `OPMODE(COMPAT,900,710`, indicating that you cannot revert the queue manager to a version prior to Version 9.0.0, and that only Version 7.1.0 new functions are available.

The syntax of **OPMODE** is as follows:

OPMODE=(Mode,*VerificationLevel*)

OPMODE specifies the operation mode of the queue manager.

LTS The default setting of **OPMODE** for IBM MQ Version 9.0.0 is `OPMODE=(COMPAT,900)` .

CD The only valid setting of **OPMODE**, for a Continuous Delivery (CD) release of IBM MQ, is `OPMODE=(NEWFUNC,90x)`. For example, at IBM MQ Version 9.0.1, you must specify `OPMODE=(NEWFUNC,901)`.

Mode

Specifies the requested operation mode. The values are as follows:

COMPAT

The queue manager runs in compatibility mode. Certain new functions are not available. The queue manager can be migrated back to an earlier release.

LTS

Important: This value is valid only for a Long Term Support (LTS) release of IBM MQ.

NEWFUNC

All new functions provided in this level of code are available. The queue manager cannot be migrated back to an earlier release.

VerificationLevel

VerificationLevel is a Version.Release.Modification (VRM) code, without punctuation; 900, for example.

The value of *VerificationLevel* ensures that the **CSQ6SYSP** parameters are coded for use with the level of **CSQ6SYSP** macro being compiled. If *VerificationLevel* does not match the VRM level of SCSQMACS used for **CSQ6SYSP**, then a compile-time error is reported. The *VerificationLevel* is compiled into the parameter module.

LTS At queue manager start up, if the *VerificationLevel* does not match the release level of the queue manager, for an LTS release, then COMPAT mode is forced.

CD For a queue manager started at a CD release, if *VerificationLevel* refers to a CD release, but does not match the release level of the queue manager, OPMODE=(NEWFUNC,*vrm*) takes effect, where *vrm* is the CD release level of the queue manager.

For example, a Version 9.0.2 queue manager started with OPMODE=(NEWFUNC,901), behaves as if OPMODE=(NEWFUNC,902) had been specified.

If *VerificationLevel* refers to an LTS release, the queue manager will not start at a CD release.

The intent of the *VerificationLevel* parameter is to avoid inadvertent and irreversible setting of OPMODE to NEWFUNC. The mistake might occur when migrating to a newer version of IBM MQ using **CSQ6SYSP** statements prepared for an older version of the queue manager. It might also occur using a **CSQ6SYSP** parameter module built with an older version of the SCSQMACS macros.

If you need assistance to revert to an earlier version of IBM MQ, contact your IBM support center.

Related information:

z/OS: Switching from new function mode to compatibility mode
The availability of new functions and backward migration for IBM MQ for z/OS is controlled by the **OPMODE** parameter in the **CSQ6SYSP** macro. You should be aware of the implications of switching from new function mode to compatibility mode, that is, switching from OPMODE=(NEWFUNC,800) or OPMODE=(NEWFUNC,900) to OPMODE=(COMPAT,800) or OPMODE=(COMPAT,900).

Using CSQ6SYSP

OPMODE restrictions by version on z/OS: z/OS

The availability of some new functions and backward migration for IBM MQ for z/OS is controlled by the **OPMODE** parameter in the **CSQ6SYSP** macro. The **OPMODE** parameter determines whether you can use selected new functions, before you commit to staying at a given release. The functions and capabilities that are restricted in the different versions of the product are listed here.

IBM WebSphere MQ Version 7.1 on z/OS

Table 92. Functions and capabilities in Version 7.1 on z/OS

Function	Reference
CFLEVEL(5) which is a prerequisite for Shared Message Data Sets (SMDS) and custom offload rules	DEFINE CFSTRUCT Planning your coupling facility and offload storag
Support for the X'3C' OTMA protocol messages, which report IMS health status	The IMS bridge
Resilience to coupling facility connectivity failures	Shared queue recovery

IBM MQ Version 8.0 on z/OS

Table 93. Functions and capabilities in Version 8.0 on z/OS

Function	**Reference**
Buffer pools can be located above the bar	DEFINE BUFFPOOL
Multiple cluster transmission queues	Clustering: Planning how to configure cluster transmission queues Working with cluster transmission queues and cluster-sender channels Planning how you use multiple cluster transmission queues
Increased maximum addressable log range	Larger log Relative Byte Address
Configurable certificates for individual channels using CERTLABL	Certificate label (CERTLABL)
Using host names, restricting the certificate issuer, and checking client credentials in CHLAUTH rules	SET CHLAUTH

Note: CD These functions are also restricted by OPMODE in IBM MQ Version 9.0

Related information:

Using CSQ6SYSP

Switching from new function mode to compatibility mode on z/OS: z/OS

The availability of new functions and backward migration for IBM MQ for z/OS is controlled by the **OPMODE** parameter in the **CSQ6SYSP** macro. You should be aware of the implications of switching from new function mode to compatibility mode, that is, switching from `OPMODE=(NEWFUNC,800)` or `OPMODE=(NEWFUNC,900)` to `OPMODE=(COMPAT,800)` or `OPMODE=(COMPAT,900)`.

CD

Note: Backwards migration is not supported for a Continuous Delivery (CD) release. Queue managers running a CD release of IBM MQ must be started with **OPMODE**=(`NEWFUNC,90x`) where `x` is the modification level.

Switching from OPMODE=(NEWFUNC,800) or OPMODE=(NEWFUNC,900) to OPMODE=(COMPAT,800) or OPMODE=(COMPAT,900)

When you switch from `OPMODE=(NEWFUNC,800)` or `OPMODE=(NEWFUNC,900)` to `OPMODE=(COMPAT,800)` or `OPMODE=(COMPAT,900)`, certain new Version 8.0 functions will no longer be available. This will cause the following conditions to occur:

- If the BSDS has been converted to version 2, the queue manager will not be able to access the BSDS when it is started in compatibility mode. This means that the queue manager fails to start and terminates with reason code 00D10120.
  - You can find the BSDS version by running the print log map utility (**CSQJU004**).
- Any buffer pools with an ID greater than 15 are marked as suspended. This means that these buffer pools cannot be used, deleted, or altered until the queue manager starts in new function mode again. Information about the buffer pools is kept in checkpoint log records until the queue manager starts in new function mode again.
  - Any page set that uses a suspended buffer pool is also suspended. Information about the suspended page set is also kept in checkpoint records.
  - While a page set is suspended, any messages in the page set are unavailable. An attempt to use a queue or topic which uses the suspended page set results in an `MQRC_PAGESET_ERROR` message.

- While it is suspended, a page set can be associated with a different buffer pool by using the FORMAT function of the utility program CSQUTIL, specifying `TYPE(REPLACE)`. You can then issue a **`DEFINE PSID`** command to bring the page set back into use with a different buffer pool.
  -

    Note: All units of recovery that involved the suspended page set, except units that are indoubt, will have been backed out by the queue manager when the page set was last used. Indoubt units of recovery can be resolved when the page set is again in use by the queue manager.
- Any buffer pools with an ID of 15 or less that have their **`LOCATION`** attribute set to `ABOVE`, have the **`LOCATION`** attribute switched to `BELOW` and their **`PAGECLAS`** attribute set to 4KB and the buffer pool size is set to 1000 pages.
- Any cluster-sender channels that have been configured to use a transmission queue other than SYSTEM.CLUSTER.TRANSMIT.QUEUE fail to start with message CSQX295E. To allow these channels to start, you need to perform the following actions:
  - Change the default cluster transmission queue configuration of the queue manager, so that all cluster-sender channels default to use the transmission queue SYSTEM.CLUSTER.TRANSMIT.QUEUE. You can do this by changing the value of the **`DEFCLXQ`** queue manager attribute to `SCTQ`.
  - Identify any manually defined transmission queues that have a non blank cluster channel name attribute value, by using the following command:
    ```
    DISPLAY QLOCAL(*) WHERE(CLCHNAME NE ' ')
    ```
    Change the cluster channel-name attribute value of these queues to blank.
- No inbound channels will be allowed to start if any channel authentication records have been created with a host name specified in their **`ADDRESS`** attribute. Message CSQY344E is issued for each channel authentication rule that uses restricted function, if this condition occurs.
- Defining channel authentication (**`CHLAUTH`**) with the *CHKCLNT* attribute, requires the queue manager to be running in `NEWFUNC` mode.

If you need assistance to revert to an earlier version of IBM MQ, contact your IBM support center.

Related information:

Using CSQ6SYSP

DISPLAY SYSTEM

Upgrade and migration of IBM MQ on z/OS: z/OS

You can install new releases of IBM MQ to upgrade IBM MQ to a new release, or version level. Multiple installations at the same or different levels can coexist on the same z/OS instance. Running a queue manager at a higher level requires migration.

From IBM MQ for z/OS Version 9.0, the way you upgrade the systems in your enterprise has changed. See IBM MQ Release Types for more information.

Important: LTS Backwards migration is possible only from a Long Term Support (LTS) release.

When you install a new VRM level of IBM MQ on z/OS using SMP/E, it creates a set of IBM MQ libraries. The libraries for different VRM levels of IBM MQ can coexist on the same instance of z/OS. You can then run different queue managers against different release levels of IBM MQ on the same z/OS instance.

If you start a queue manager running on a later release level, then migration of the queue manager to that release level is required. Even if the difference is only in the modification level, some migration might be required. The migration tasks that you must perform to migrate from one version to another are documented in “Planning to migrate to the latest release on z/OS” on page 772; see also “Changes that affect migration” on page 626.

From Version 7.0.1, after you have fully migrated a queue manager to a new version or release, reverse migration is not possible. For Version 7.0.1 and later versions, you have control over when migration takes place, using a new **CSQ6SYSP** parameter, **OPMODE**; see “OPMODE on z/OS” on page 781. If your queue manager is on Version 7.0 or earlier, you can revert to an earlier release. You might have to contact your IBM support center for a backward migration PTF.

Using **OPMODE**, you can migrate all your existing applications to the new release level, and still be able to revert to the previous release level. Once you start changing applications, or adding applications that use new function, you cannot revert to the previous level of the product. **OPMODE** applies to migration from Version 6.0 to Version 7.0.1 onwards.

OPMODE gives you the option of enforcing a two-stage migration process:
1. Regression test your existing applications.
2. Develop new applications, and change existing applications, to use the new function in the release.

The strategy for upgrading queue managers at Version 6.0 or later is as follows:
1. Apply the coexistence and backward migration PTFs to all the queue managers you are going to upgrade. After applying the PTFs, you can run queue managers of different levels in the same queue-sharing groups. You can also reverse the migration of a queue manager back to your current level.
2. Upgrade the first queue manager.
3. Check all your existing applications run correctly on this queue manager.
4. Bring all the queue managers in a queue-sharing group up to the new level, and check that existing applications continue to work correctly.
5. Change the setting of **OPMODE** so that applications can use new function on all the queue managers in the queue-sharing group.

   Note: Step 5 is the point of no return. You can no longer run that queue manager at the previous level of the product.
6. To enable new Version 7.1 or later, function, restart all queue managers within the queue-sharing group.

The coexistence and backward migration PTFs have two distinct purposes:[10]
1. To allow queue managers at the earlier release level to coexist with ones at the later release level. In particular for queue managers to coexist in the same queue-sharing group.
2. To handle queue manager data and logs formatted using the data definitions of the later release.

Characteristics of different types of upgrade on z/OS

When you upgrade from one release to another on z/OS, the impact of the change depends on the extent of the change in VRM level. The VRM codes are explained in “The version naming scheme for IBM MQ for z/OS” on page 569.

Note that migration is required if the version, release, or modification number changes.

LTS From Version 7.0.1, all upgrades from Version 6.0 or later to a Version 9.0 Long Term Support (LTS) release are reversible if the **OPMODE** has not been set to NEWFUNC.

10. Coexistence and backward migration changes might be shipped as a single or multiple fixes.

CD Upgrades to a Continuous Delivery (CD) release are not reversible.

Related concepts:

"The version naming scheme for IBM MQ for z/OS" on page 569
On IBM MQ for z/OS, releases have a three digit Version, Release, and Modification (VRM) code. To run a queue manager at a different VRM level, you must migrate the queue manager, its applications, and the environment in which it runs. Depending on the migration path, the migration might require more or less effort.

"Queue manager coexistence" on page 664
Queue managers, with different names, can coexist on any server as long as they use the same IBM MQ installation. On z/OS, UNIX, Linux, and Windows, different queue managers can coexist on the same server and be associated with different installations.

"Backward migration to earlier supported releases of IBM MQ for z/OS" on page 789
After installation of a new release of IBM MQ for z/OS, you carry out queue manager migration by stopping the queue manager, which is running with the prior release of code, and restarting the queue manager using the new release of code.

"Migration methods on IBM MQ for Multiplatforms" on page 653
There are three main methods of migrating from one release to another: Single-stage migration (called a slip installation on IBM i), side-by-side migration, and multi-stage migration. Multi-stage migration is not an option for IBM i.

Order of migration tasks on z/OS: z/OS

Read this information that shows an overall migration plan for your system, together with the order in which you must undertake the tasks.

Before you begin

Read the information in Clustering: Best practices and Clustering: Topology design considerations to understand repositories.

About this task

The tables within this topic show the tasks required in each part of the process to migrate an overall system, and the order in which these tasks must be done.

Notes:

- You must perform the tasks in the following order:
  1. Overview of migration
  2. Migrating your system to the next release

Queue managers in a queue-sharing group (QSG) and queue managers in a cluster can be migrated in parallel, but at any time there should be enough queue managers working in the QSG, and cluster, to ensure that your business can operate satisfactorily while a staged migration takes place.

If there are queue managers in clusters, ensure that you migrate the full repository queue managers before migrating any partial repository queue managers, so that the cluster always has a full picture of the current cluster architecture.

Table 94. Overview of migration on z/OS

Task	For your own use
1. Make your configuration ready for migration	
2. Install the new code	
3. Server application migration	

Table 95. Migrating your system to the next release on z/OS

Task	For your own use
Full repositories	
Overview - Migrate one full repository on IBM MQ for z/OS in a QSG	
4. Migrate DB/2 tables and New CF structure definitions	
5. Migrate the queue manager	
6. Migrate other full repositories on IBM MQ for z/OS, if any	
7. Full repositories not in a QSG	
8. Full repositories on other platforms	
Other queue managers	
Overview - Migrate partial repositories on IBM MQ for z/OS, in a QSG	
9. Migrate DB/2 tables and New CF structure definitions if not yet done	
10. Migrate one queue manager	
11. Test the migration and upgrade the other queue managers	
12. Migrate the partial repositories not in a QSG	
Migrate queue managers not in a queue-sharing group or cluster on IBM MQ for z/OS	
13. Migrate partial repositories on other platforms	
14. Migrate queue managers not in a cluster on other platforms	
Testing and using new function; see “Post migration tasks” on page 803	
15. Regression tests	
16. Use the new function available to you	

You can upgrade the client libraries at any time during the process. As a final task, recompile the clients using new functions and deploy.

Results

You have migrated your system to another release.

Migrating from earlier unsupported releases of IBM WebSphere MQ for z/OS: z/OS

You must take into consideration whether you are upgrading a production system, or a test system, before you undertake the migration process.

Production systems

For production systems you must, firstly, migrate the unsupported release of IBM MQ to IBM WebSphere MQ Version 7.1, following the instructions given in the documentation for that release. For more information about where to find the documentation for older versions of the product, see Documentation for older versions of IBM MQ

You can then migrate to IBM MQ Version 9.0 following the instructions in this section.

Important: Ensure that your system is stable at Version 7.1, before migrating to Version 9.0, so that you have a system to revert to, if necessary.

Test systems

For a test system, it might be appropriate to migrate directly to IBM MQ Version 9.0 instead.

Before you begin the migration, take complete backups of your system, to ensure that you can restart from backups if you need to use the old release again.

IBM MQ Version 9.0 migrates IBM MQ objects and messages during the first startup at Version 9.0.

New attributes added to objects in IBM WebSphere MQ Version 7.0 and IBM MQ Version 9.0 releases are set to their default values.

Important: After you have migrated to IBM MQ Version 9.0 using this method, you cannot revert to the original version.

You can restart a queue manager, at the original version, using the full set of backups taken before migration. However, note that any changes you make to the system after the backups were taken, or while running at Version 9.0, will be lost.

Backward migration to earlier supported releases of IBM MQ for z/OS: z/OS

After installation of a new release of IBM MQ for z/OS, you carry out queue manager migration by stopping the queue manager, which is running with the prior release of code, and restarting the queue manager using the new release of code.

Maintenance in a queue-sharing group

In a queue-sharing group, individual queue managers can be migrated forwards to IBM MQ Version 9.0, while those that remain at either IBM WebSphere MQ Version 7.1.0 or IBM MQ Version 8.0.0 can continue to function. This allows you to upgrade queue-sharing group queue managers to Version 9.0 at different times, maintaining the high availability of the queue-sharing group.

The function required to enable lower level queue managers to tolerate Version 9.0 additions to QSGDISP(GROUP) and QSGDISP(SHARED) objects is incorporated in the same authorized program analysis reports (APARs) which provide backward migration capability.

Code levels supported

Migration support is provided from both IBM WebSphere MQ Version 7.1.0 and IBM MQ Version 8.0.0 to IBM MQ for z/OS Version 9.0.

The backward migration APARs are PI64465 for IBM WebSphere MQ Version 7.1.0, and PI64466 for IBM MQ Version 8.0.0.

Important: LTS PTFs for these APARs must be applied on IBM WebSphere MQ Version 7.1.0 or IBM MQ Version 8.0.0 prior to attempting to fall back from IBM MQ for z/OS Version 9.0.0 Long Term Support (LTS) release.

CD Backwards migration is not supported for a Continuous Delivery (CD) release.

PTFs for these APARs are the *Migration and Toleration PTFs for Version 9.0* described in Planning for migration to the latest release.

Service has been discontinued for versions of the product prior to IBM WebSphere MQ Version 7.1.0. No backward migration capability is available for these versions.

CD The IBM MQ for z/OS Version 9.0.0 and IBM MQ for z/OS Version 9.0.1 early code installed in the link pack area (LPA) is downward compatible. The code supports queue managers running at IBM WebSphere MQ Version 7.1.0 and IBM MQ Version 8.0.0.

Once updated to the Version 9.0 level, and the queue manager subsystem refreshed using the REFRESH QMGR TYPE(EARLY) command, the early code need not be changed for any subsequent forward or backward migration activity

Message

```
CSQ3111I <cpf> CSQYSCMD - EARLY PROCESSING PROGRAM IS V9.0 LEVEL 008-000
```

is displayed during startup in the queue manager joblog and indicates that the queue manager is using the correct level of early code.

Limitations and restrictions

IBM MQ for z/OS Version 9.0 uses a migration switch to support backward migration by preventing use of certain new functions, that cannot be backward migrated, until the installation confirms that backward migration is no longer required.

The migration switch is configured through a change to ZPARM using the OPMODE parameter of CSQ6SYSP.

LTS While **OPMODE** is set to COMPAT, it is possible to backward migrate from a Long Term Support (LTS) release, although certain new functions are not available. Once **OPMODE** is set to NEWFUNC, all new functions are available, but it is no longer possible to perform backward migration.

CD Backwards migration is not supported for a Continuous Delivery (CD) release. Queue managers running a CD release of IBM MQ must be started with **OPMODE**=(NEWFUNC,90x). For example, a Version 9.0.1 queue manager must be started with **OPMODE**=(NEWFUNC,901).

The MQSC command DISPLAY SYSTEM command displays three values, the operation mode, either COMPAT or NEWFUNC, and two version numbers. The first version number indicates to which version of IBM MQ for z/OS you can fall back to. The second version number indicates level of new functions that are available

When the operation mode is COMPAT, then the version number indicates to which version of IBM MQ for z/OS you can fall back.

The value of OPMODE displayed during startup in message CSQY101I reflects the operation mode requested using ZPARM. Queue manager initialization evaluates the requested operation mode in combination with local state and other members of the queue-sharing group, to determine the actual operation mode displayed with DISPLAY SYSTEM.

You cannot backward migrate a queue manager, newly created at Version 9.0, to a prior release. A queue manager migrated forward to Version 9.0 *remembers* where it was migrated from, and it is only possible to fall back to that *remembered* prior version.

Certain connection types (IMS, BATCH and RRSBATCH used by WAS and Db2 stored procedures) allow an application to connect to multiple queue managers concurrently. If required, these queue managers can be running different levels of IBM MQ code. In such a scenario, the adapter code (usually referenced through a STEPLIB DD statement or environment variable) must be loaded from libraries corresponding with the highest level of the queue managers connected. This ability for the adapter code to support connections to older queue managers means that in a backward migration scenario it is possible to just restart the MSTR and CHIN procedures with the back level code, and not change connecting jobs.

The operations and controls ISPF panels, CSQOREXX, from IBM MQ for z/OS Version 9.0, are able to connect to and administer queue managers from a prior release. However, the ISPF panels from lower releases are not able to connect to IBM MQ for z/OS Version 9.0. When migrating, or during fall back, either use the same version ISPF panels as the level of code the queue manager is running, or use CSQOREXX from the higher release of code. In a mixed level queue-sharing group, the IBM MQ for z/OS Version 9.0 panels must be used to administer Version 8.0.0 or Version 7.1 queue managers, as ISPF panels from earlier releases do not tolerate responses from any Version 9.0 queue managers.

Related reference:

"Switching from new function mode to compatibility mode on z/OS" on page 784
The availability of new functions and backward migration for IBM MQ for z/OS is controlled by the **`OPMODE`** parameter in the **`CSQ6SYSP`** macro. You should be aware of the implications of switching from new function mode to compatibility mode, that is, switching from `OPMODE=(NEWFUNC,800)` or `OPMODE=(NEWFUNC,900)` to `OPMODE=(COMPAT,800)` or `OPMODE=(COMPAT,900)`.

Preparing to migrate a single IBM MQ for z/OS queue manager: z/OS

Follow the steps to prepare a single IBM MQ queue manager on z/OS for migration.

About this task

To prepare to migrate an IBM MQ queue manager on z/OS, you need to carry out the detailed steps in this topic, using the links within this overview.

1. Make your existing queue manager ready for migration; see Step 1
2. Install the new code and make target libraries available to all MVS systems that are running queue managers, and grant access; see Step 2.
3. Perform a back up operation of each queue manager in your enterprise; see Step 3.
4. Review definitions of the user IDs for the queue manager(MSTR) and channel initiator (CHIN) address spaces; see Step 4
5. Restart your IBM MQ systems; see Step 5.

6. Review pageset zero usage before migration; see Step 6.
7. Migrate your Db2 tables, and repeat this step for each queue-sharing group (QSG), if your enterprise uses QSGs; see Step 7
8. Add a new coupling facility (CF) structure definition and repeat this step for each QSG, if your enterprise uses QSGs; see Step 8.
9. Consider the migration of your server applications; see Step 9
10. Configure Advanced Message Security (AMS); see Step 10

.

Procedure

1. Make your IBM MQ configuration ready for migration.
   a. Refer to the Preventive Service Planning (PSP) bucket for your version of IBM MQ; see PSP Buckets - How to find them on Web.
   b. Apply the migration and toleration PTFs to the version of the IBM MQ code that your enterprise uses; see IBM MQ Support, Migration PTFs.

      Note that the “migration and toleration” PTFs are also known as the “backward migration and coexistence” PTFs; they are the same PTFs.

      If you are unsure which migration PTFs you require, run the following command SMP/E:

      CD

      ```
      REPORT MISSINGFIX ZONES(mqtgtzone) FIXCAT(IBM.Coexistence.MQ.V9R0M0)
      ```

      See FIXCAT and IBM MQ Migration Installation for further information.

      Attention: If a PTF requires a rebind of Db2 plans, the PTF is shipped with ++HOLD(ACTION), indicating the need for this process. In such a case, see Migrating Db2 tables to bind the plans before starting migration.

      Other FIXCAT categories are listed in IBM Fix Category Values and Descriptions.

      There is an additional category of `TargetSystem-RequiredService.MQ.V9R0M0` enabling other products to run with IBM MQ Version 9.0.
2. Install the new code and make target libraries available to all MVS systems that are running queue managers, and grant access. You must carry out the following procedure for each MVS system.
   a. Copy the IBM MQ target libraries to the system, and install the early code for the new version (once for each MVS system). Activate the code for all of the queue managers on each MVS system that is running queue managers.

      This updates the LPA. See Update the z/OS link list and LPA for more information.
   b. APF authorize the load libraries and grant access to the data sets using your external security system. See APF authorize the IBM MQ load libraries for more information.
   c. Copy the file system zFS and mount it read only.

      You only need zFS, or older HFS, if the IBM MQ for z/OS Unix System Services Component is installed. See the Program Directory for further information.

   Refresh all the queue managers so that they use the new early code using the command REFRESH QMGR TYPE(EARLY). See REFRESH QMGR for more information.
3. Perform a back up operation for each queue manager in your enterprise, so that you have a `before` copy of all objects and JCL before you make any changes. This makes rolling back to the current system easier, if you require to do so.
   a. Back up your IBM MQ defined objects, for example using CSQUTIL COMMAND MAKEDEF(..) See Using the COMMAND function of CSQUTIL for more information.
   b. Back up:
      - The MSTR and CHINIT started procedure jobs

- The Initialization input data sets used in the CSQINP1 and CSQINP2 concatenations
- The system parameter module (ZPARM) libraries
- Other tasks as necessary.

Note: You might also make a back up of page sets, BSDSs, and active logs as a fallback option. See How to back up and recover pagesets for more information on backing up IBM MQ resources.

4. Check that MSTR and CHIN address spaces run under user IDs that have OMVS segments defined, with a valid UID, to enable calling Unix System Services (USS).
5. Restart your IBM MQ system to run with the migration and toleration PTFs.
   a. Restart the queue managers and monitor the whole system in your enterprise closely to ensure that there are no issues. Depending on the size and complexity of your enterprise this can take a considerable length of time, so you must plan for this in your migration schedule.
6. Review the usage of pageset 0. Note that you can ignore this step if your enterprise is already using IBM WebSphere MQ Version 7.1. Issue the operator command `/cpf DISPLAY USAGE PSID(0)`, where **`cpf`** is the command prefix for the subsystem of the queue manager, to get a report on pageset 0 usage.

   The size of queue definitions increased in IBM WebSphere MQ Version 7.1. During migration to this version, or later versions of the product from an earlier version of the product, queue definitions stored on pageset 0 are rewritten.

   The rewrite is carried out as a single transaction when the queue manager is first migrated to IBM WebSphere MQ Version 7.1, or later.

   Ensure that there is enough space available on pageset 0 to create a copy of the queue definitions while migration is taking place. Typically, 60% free space on pageset 0 before migration is sufficient. However, the use of `EXPAND(SYSTEM)` on the pageset definition allows for automatic expansion as required.

   If there is insufficient space on pageset 0 during migration, the queue manager abends with completion code `X'5C6'` and reason code `X'00C91900'`.
7. Migrate your Db2 tables for each Db2 data sharing group.

   You must do this for each Db2 data sharing group, as multiple QSGs can use the same Db2 tables.

   You can use IBM provided samples shipped in the new version of the product to perform this task. Some Db2 table definitions are updated, and some new Db2 tables are created for the migrated version of the queue manager.

   Notes:

   a. You must have applied the migration and toleration PTFs to all the queue managers, before migrating the Db2 tables.
   b. Every queue manager in the queue-sharing group needs to be restarted at the current release, with the PTFs applied.
   c. At no stage is an outage of the entire queue-sharing group required.
   d. Migrate your Db2 tables.

      If the jobs described fail because of a Db2 locking problem, it might be due to contention for a Db2 resource. Locking is more likely, if the system is being heavily used. Resubmit the job later, preferably when the system is lightly used or quiesced.

      See steps 5 and 6 of Set up the Db2 environment.
   e. Use the `CSQ45*` jobs in the newest *`thlqual`*`.SCSQPROC` supplied with the version of the product to which you are migrating.

      Note that the JCL to use depends on the highest version of IBM MQ in the Db2 tables.

Attention: CD If the Db2 tables have IBM MQ Version 9.0 queue managers, ignore the preceding steps, b and c.

1) If the Db2 tables have IBM WebSphere MQ Version 7.1 queue managers, use `CSQ4571T`. If the Db2 tables have IBM WebSphere MQ Version 7.0 queue managers, use `CSQ4570T`.
2) Customize the `CSQ45*` sample.

   The header information in `CSQ45*` describes how to customize the sample.
3) Run the customized `CSQ45*` job.
4) Customize the `CSQ45BPL` and `CSQ45GEX` samples, in *`thlqual`*`.SCSQPROC`

   The header information in `CSQ45BPL` and `CSQ45GEX` describes how to customize the samples.
5) Run the customized jobs, `CSQ45BPL` and `CSQ45GEX`.

   If you need to rebind the plans or packages, see Step 1 on page 806 in Migrating queue-sharing groups from a previous release for further information.

f. If you have multiple QSGs in the same data sharing group (DSG) you need to check each queue-sharing group to see if each member meets its migration criteria. Use sample JCL `CSQ45MQS` in conjunction with `CSQ4571T`.

   See the JCL header description for further information.

8. Add the new coupling facility (CF) definition. Repeat this step for each QSG. Note that you can ignore this step if your enterprise is already using IBM WebSphere MQ Version 7.1.

   Starting with IBM WebSphere MQ Version 7.0.1, a new CF structure is required; see Set up the coupling facility for information on how to add such a definition.

   The correct process to migrate SYSTEM.QSG.CHANNEL.SYNCQ, from a normal application CF structure, to system CF structure CSQSYSAPPL structure is:

   a. Stop the channel initiator (CHINIT) on all queue-sharing group queue managers, so that no channels are running.
   b. Copy the messages in SYSTEM.QSG.CHANNEL.SYNCQ to a temporary data set, using CSQUTIL COPY.
   c. Delete SYSTEM.QSG.CHANNEL.SYNCQ from the repository.
   d. Define SYSTEM.QSG.CHANNEL.SYNCQ with CFSTRUCT(CSQSYSAPPL). As this is a shared queue, it only needs to be defined once per QSG. Note that you can define this queue from any queue manager within the QSG.
   e. Reload the SYNCQ messages from the temporary data set, back to the newly defined shared queue, using CSQUTIL LOAD.
   f. Perform the other migration steps, and then restart CHINIT to make the changes taking effect.

9. Migrate server applications.

   Java or JMS applications running on the same host with IBM MQ connect to queue managers in bindings mode. This is a cross-memory connection. In this mode, applications need to update their STEPLIB concatenations, so that they can always load the highest version IBM MQ library in the system.

   Note, that if a z/OS Java or JMS application is running under WebSphere Application Server, the application can use client mode as an alternative to bindings mode.

   IBM MQ libraries include:

   thlqual.SCSQANLx
   This library contains error message information for your national language. The letter 'x' represents the letter for your national language.

   thlqual.SCSQAUTH
   This library contains the code that the applications use.

   Server applications for IBM MQ can include:

   - Batch applications

- Control panels in ISPF
- IMS
- Interactive problem control system (IPCS)
- RRS adapter, including Db2 stored procedures.
- TSO
- Additionally, WebSphere Application Server for z/OS, IBM Integration Bus, and CICS.

a. You can use the "TSO ISRDDN ENQ ' `thlqual`.SCSQANLE'" command, replacing `thlqual` with the High Level Qualifier for your installation, to check which jobs are running with the specified library. You can then modify them accordingly.

b. Update STEPLIB in the application JCL, and refer to the new IBM MQ libraries.

c. Restart these applications. For further information, see:
   - Set up Batch, TSO, and RRS adapters
   - Setting up the IMS adapter
   - Set up the operations and control panels
   - Include the IBM MQ dump formatting member

d. Migrate other software, such as WebSphere Application Server, IBM Integration Bus, or CICS to use the version of IBM MQ that you need.
   - CICS

     Update the IBM MQ libraries in the `STEPLIB` and `DFHRPL` concatenations of your CICS region JCL and restart CICS.

     Up to, and including CICS 3.2, the connection between IBM MQ and CICS is provided by IBM MQ. You must change the `SCSQCICS` and `SCSQAUTH` libraries in the `DFHRPL` concatenation provided by IBM MQ.

     After CICS 3.2, the connection between IBM MQ and CICS is provided by CICS libraries. Update the libraries, if you are using CICS Transaction Server for z/OS Version 3.2 or later. Without this change, you are not able to use the most recent IBM MQ features. You must change the `SCSQCICS` library in the `DFHRPL` concatenation provided by IBM MQ, and also the `STEPLIB` concatenation.

     Create separate CICS started procedure JCL. For each CICS region that is connected to an IBM MQ queue manager, ensure that there is a separate CICS started procedure JCL.

     This ensures that the modification of reference to a certain version of IBM MQ libraries in the CICS started procedure JCL only has impact for that single CICS region. In this way you can migrate one queue manager, and only the CICS region or regions connected to it, which makes staged migration possible.

     CICS `STEPLIB` has `thlqual.SCSQAUTH`, and `DFHRPL` has `thlqual.SCSQCICS`, `thlqual.SCSQLOAD`, and `thlqual.SCSQAUTH`. For more information, see Setting up the CICS- IBM MQ adapter.
   - WAS for z/OS

     If you are running in an application server environment where a bindings connection is being used, you need to update the `WAS STEPLIB` with IBM MQ libraries.

     See IBM MQ libraries and the WebSphere Application Server for z/OS STEPLIB for further information.

     You also need to configure the IBM MQ messaging provider with native libraries from the new version of the IBM MQ installation; see Configuring the IBM MQ messaging provider with native libraries for further information.

     Use the latest level of native libraries in USS.

   Note that you can make use of a DFP ALIAS for convenience. Create data set aliases such as MQM.SCSLOAD, and reference them in JCL. Map the aliases to the real data sets, such as MQM.V700.SCSLOAD or MQM.V710.SCSLOAD.

Change the aliases to switch between the two sets of target libraries. With the aliases, you can start applications or the queue manager when moving to a new release of IBM MQ without changing the STEPLIB JCL.

10. Configure Advanced Message Security (AMS).

    If the queue manager is configured to use Advanced Message Security (AMS), perform the steps in the Preparing to migrate Advanced Message Security section of the Migrating Advanced Message Security topic.

Results

You have prepared your IBM MQ queue manager on z/OS for migration.

What to do next

Follow the instructions in "Migrating a single IBM MQ z/OS queue manager to the next release of the product" to migrate the queue manager.

Related information:

IBM MQ for z/OS Program Directory PDF files

Migrating a single IBM MQ z/OS queue manager to the next release of the product: z/OS

Carry out the instructions in this topic to migrate a single IBM MQ queue manager on z/OS,

About this task

To migrate an IBM MQ queue manager on z/OS to a different release, you need to carry out the:

- Process described in "Preparing to migrate a single IBM MQ for z/OS queue manager" on page 791
- Detailed steps in this topic, using the links within this overview.
  1. Stop or disconnect the applications; see step 1
  2. Stop the queue manager and its channel initiator; see step 2
  3. Update STEPLIB for MSTR and the channel initiator; see step 3
  4. CD Update the target version system parameter module (ZPARM); see step 4 on page 797
  5. Configure Advanced Message Security; see step 5 on page 797
  6. Review the security control of your system; see step 6 on page 797
  7. Start the queue manager; see Step 7 on page 797
  8. Optionally, revert the queue manager to a previous version; see step 8 on page 797

Procedure

1. Stop or disconnect all the applications using the queue manager (for example, CICS, IMS, or batch) and the IBM MQ channels that are connected to other queue managers.
2. Stop the queue manager and its channel initiator.
3. Update STEPLIB for MSTR and the channel initiator (CHINIT). Update the start procedure and CHINIT JCL.
   a. Update your procedure to start the queue manager.

      Change the `STEPLIB` for the queue manager to reference the new version of the libraries.

      See Create procedures for the IBM MQ queue manager.

      IBM MQ now uses z/OS memory objects above the bar for some functions. You must allow the queue manager to access storage above the bar.

Your installation might have customized the `SMFPRMxx` member of `SYS1.PARMLIB`, or the **`IEFUSI`** exit to provide a default limit for jobs using virtual storage above the 2 GB bar. Check these limits give sufficient memory for a queue manager. A reasonable starting allocation is 2 GB. The message CSQY220I displays the amount of virtual storage currently used and available.

If your installation does not have a default limit for storage above the bar, or if you want to use a different limit for your queue manager, you can provide a queue manager-specific restriction on the amount of virtual storage available above the bar for memory objects by coding a **`MEMLIMIT`** parameter on the JCL of the queue manager stored procedure, `xxxxMSTR`, for example:

```
//PROCSTEP EXEC PGM=CSQYASCP,REGION=0M,MEMLIMIT=2G
```

`MEMLIMIT` defines memory available above the bar; see Address space storage

You must allow the queue manager to access storage above the bar because IBM MQ uses memory above the bar.

If insufficient storage is available above the bar, the queue manager reports this when starting, and stops.

b. Update your procedures for the channel initiator.

   Change the `STEPLIB` for the channel initiator to reference the new level of the product libraries.

   See Create procedures for the channel initiator.

4. CD If migrating to a Continuous Delivery (CD) release for the first time, you must configure the **`OPMODE`** parameter in the system parameter module (ZPARM) to start in new function mode. For example, **`OPMODE`**`=(NEWFUNC,901)`

   Important: Do not update ZPARM at this stage when migrating to a Version 9.0.0 Long Term Support (LTS) release.

   For further details, see OPMODE.
5. If the queue manager is configured to use Advanced Message Security (AMS), perform the steps in Migrating Advanced Message Security.
6. Review your security control for queue-sharing groups, the channel initiator, and all queue managers accessing the coupling facility list structures.
7. Start the queue manager. Test that everything is working correctly and, if it is, start the channel initiator. If there is a problem starting the queue manager, consider reverting the queue manager to a previous version; see step 7.
8. If a problem occurs when starting the queue manager, you might need to consider backward migration; see Reverting a queue manager to a previous release.

Results

You have migrated your IBM MQ for z/OS queue manager to the latest release.

What to do next

Follow the instructions in "Post migration tasks" on page 803 to complete the migration process.

CSQINP1 and CSQINP2 input data sets changed on z/OS: z/OS

The CSQINP1 and CSQINP2 initialization input data sets changed in Version 7.1. The data sets include more samples and the contents of some samples have been moved to other samples. Particular changes to take note of are the commands to define queues to hold publish/subscribe state information. The commands must be in the right order.

Important changes to initialized input data sets since IBM WebSphere MQ Version 7.1 on z/OS

CSQ4INSM
Added for Advanced Message Security support

CSQ4INSG
Add one **AUTHINFO** object SYSTEM.DEFAULT.AUTHINFO.IDPWOS for connection authentication support.

Some channel objects and topic objects are modified with new attributes, for example, **STATCHL** and **CLROUTE**.

CSQ4INST
The default system subscription, SYSTEM.DEFAULT.SUB, moved from CSQ4INSG to CSQ4INST in IBM WebSphere MQ Version 7.1.

CSQ4INSX
Add one model queue SYSTEM.CLUSTER.TRANSMIT.MODEL.QUEUE for multiple cluster transmission queue support.

Review the changes, and update the customized versions you are currently using as required, if your enterprise is using IBM WebSphere MQ Version 7.0.

The queue manager uses queues to hold persistent state information about publish/subscribe. Durable subscriptions are held as messages on SYSTEM.DURABLE.SUBSCRIBER.QUEUE and retained publications on SYSTEM.RETAINED.PUB.QUEUE.

The ordering of the definitions of these queues is important. Member CSQ4MSTR of **SCSQPROC** shows the necessary ordering of the supplied definitions in the CSQINP2 concatenation. The default system subscription, SYSTEM.DEFAULT.SUB, requires SYSTEM.DURABLE.SUBSCRIBER.QUEUE that in turn requires the storage class SYSLNGLV, which is defined in CSQ4INYS. If you migrating from a previous release, and modifying customized procedures, define these resources in the following order:

1. Storage class SYSLNGLV. It requires a defined mapping to a defined page set.
2. SYSTEM.DURABLE.SUBSCRIBER.QUEUE
3. SYSTEM.DEFAULT.SUB

Note:

The changes to CSQINP1 and CSQINP2 are required in Create procedures for the IBM MQ queue manager and Customize the initialization input data sets.

z/OS Migrating a queue manager to mixed case security: z/OS

Follow these steps to migrate a queue manager to mixed case security. You review the level of security product you are using and activate the new IBM MQ external security monitor classes. Run the **REFRESH SECURITY** command to activate the mixed-case profiles.

Before you begin

1. Install a level of the security product that supports mixed case security.
2. Apply any updates required by IBM MQ.
3. Install and activate the new IBM MQ external security monitor classes.

About this task

Follow these steps to convert a queue manager to mixed case security.

Procedure

1. Copy all your existing profiles and access levels from the uppercase classes to the equivalent mixed case external security monitor class.
   a. `MQADMIN` to `MXADMIN`.
   b. `MQPROC` to `MXPROC`.
   c. `MQNLIST` to `MXNLIST`.
   d. `MQQUEUE` to `MXQUEUE`.
2. Start the queue manager.

   The queue manager `SCYCASE` attribute is set to `UPPER`.
3. Change the value of the `SCYCASE` attribute to `MIXED`.
   ```
   ALTER QMGR SCYCASE(MIXED)
   ```
4. Activate your existing security profiles.
   ```
   REFRESH SECURITY(*) TYPE(CLASSES)
   ```
5. Test that your security profiles are working correctly.

What to do next

Review your object definitions and create new mixed case profiles as appropriate, using **REFRESH SECURITY** as required to activate the profiles.

Migrating Advanced Message Security on z/OS: z/OS

Advanced Message Security for z/OS (AMS) is a separately licensed enabling product that extends IBM MQ to provide a high level of protection for sensitive data flowing through the IBM MQ network using a public key cryptography model.

In IBM MQ for z/OS releases prior to Version 8.0, AMS was provided as a separate product. This topic describes the tasks required to migrate the AMS configuration on z/OS from that used in version 7 and earlier, to that used in Version 8.0. These steps supplement those required to migrate a single IBM MQ for z/OS queue manager where AMS is not configured. AMS must be migrated at the same time as the queue manager, it is not supported to use Advanced Message Security Version 7.0.1 with IBM MQ for z/OS Version 8.0.

To enable AMS on a newly created IBM MQ for z/OS queue manager, or on a queue manager that has already been migrated to Version 8.0, see Advanced Message Security for z/OS.

Preparing to migrate Advanced Message Security on z/OS

To prepare to migrate an IBM MQ queue manager on z/OS using AMS Version 7.0.1 or earlier, you must perform the steps in this section in addition to those listed in “Preparing to migrate a single IBM MQ for z/OS queue manager” on page 791.

1. Install the Advanced Message Security for z/OS enabling product and make the target libraries available to all MVS systems that are running queue managers that will use AMS. You must carry out the following procedure for each MVS system:
   a. Copy the AMS target libraries to the system.
   b. APF authorize the thlqual.SDRQAUTH target library and grant access to this data set using your external security system, see Task 2: APF authorize the IBM MQ load libraries.
   c. Ensure the LPA contains the AMS module CSQ0DRTM, see Task 3: Update the z/OS link list and LPA.
   d. Ensure the program properties table (PPT) contains an entry for CSQ0DSRV, see Task 4: Update the z/OS program properties table.
2. For each queue manager, set up the started task user for the AMS address space. In AMS Version 7.0.1 two address spaces are used, one for the main task and another for the data services task. In Version 8.0 these are combined in to a single address space called `qmgrAMSM`. Either set up a new user for the Version 8.0 AMS address space, or grant additional authorities to one of the existing AMS started task users. See Task 25: Set up the started task user Advanced Message Security for information on how to set up the started task user. If you do not use the existing data services address space user you will need to replicate the **drq.ams.keyring** key ring for the user ID associated with the Version 8.0 `qmgrAMSM` address space. See Using certificates on z/OS for information on how to set up the AMS key ring.

Migrating Advanced Message Security on z/OS

To migrate an IBM MQ queue manager on z/OS using AMS Version 7.0.1 or earlier, before restarting the queue manager you must perform the steps in this section in addition to those listed in “Migrating a single IBM MQ z/OS queue manager to the next release of the product” on page 796.

1. Take a copy of the `qmgrAMSM` task for Advanced Message Security (AMS) Version 7.0.1, in case you need to revert to your previous system.

   See “Backward migration of Advanced Message Security on z/OS” on page 801 for more information.
2. Configure the queue manager to use AMS by updating the system parameter module to set SPLCAP(YES) using CSQ6SYSP, see Task 17: Tailor your system parameter module and Using CSQ6SYSP.
3. Create or update the started task procedure for the `qmgrAMSM` address space, see Task 24: Create procedures for Advanced Message Security.

Post migration tasks for Advanced Message Security on z/OS

After you have migrated an IBM MQ queue manager on z/OS that uses AMS you must perform the following tasks.

1. In Version 8.0, the AMS address space is started and stopped automatically by the queue manager. If you have automation to manage the main task and data services task for AMS Version 7.0.1, or earlier, this should be removed. You must also review any automated console commands for AMS because some have changed in version 8.
2. Delete the started task procedures for the Version 7.0.1 data services task and the Version 7.0 main task if these were not called `qmgrAMSM`.

Backward migration of Advanced Message Security on z/OS

If you are an AMS user, and you backward migrate your queue manager from Version 8.0 to a version 7 release, additional actions are required to revert AMS to version 7.

Considerations when migrating

You should ensure that your previous setup is in place and that tasks Updating the z/OS LPA to Updating your system DIAG member have been carried out.

Ensure that the user ID associated with the version 7 data-services address spaces has access to `drq.ams.keyring`, and that `drq.ams.keyring` has the same connected certificates as the Version 8.0 `qmgrAMSM` user ID.

Performing the migration

When you have completed the previous tasks, you can migrate your queue manager backwards in the normal way.

Manually start, or reintroduce automation for starting, the AMS main and data services address spaces.

See Starting Advanced Message Security for further information.

Reverting a queue manager to a previous release on z/OS: z/OS

After migrating to IBM MQ for z/OS Version 9.0.0 Long Term Support (LTS) release, from either Version 7.1.0 or Version 8.0.0, you can backward migrate, or fallback, to the version you were using prior to migration. Backward migration Program Temporary Fixes (PTFs) are available for both Version 7.1.0 and Version 8.0.0. Backwards migration is not supported for a Continuous Delivery (CD) release.

Before you begin

In general, after fallback to IBM WebSphere MQ Version 7.1, new attributes of IBM MQ objects introduced at Version 9.0 will be removed. The APAR that supplies these PTFs, documents specific information relating to fallback to IBM WebSphere MQ Version 7.1 or to IBM MQ Version 8.0.0.

Switching back to running a queue manager with the target libraries of a previous version is possible if **`DISPLAY`** `SYSTEM` returns `COMPAT,vrm,nnn` where `vrm` is the level of the previous version.

7 or 8	is the version number `v` of the product.
r	is the release number of the product.
m	is the modification number of the product.

If it does, you can go back to using your customization and startup procedure for the queue manager from that version:

- The queue manager compatibility level must be `710` or `800`. This will be the case if the queue manager has never been started with OPMODE set to `(NEWFUNC,900)`.
  - **`DISPLAY`** `SYSTEM` returns OPMODE `COMPAT,710,nnn` or OPMODE `COMPAT,800,nnn`.
- Before migrating your queue manager to the latest version, with that version of target libraries, you applied all the migration and toleration PTFs to the queue manager on your previous version. The queue manager then started successfully with those PTFs at that previous version. This is a requirement before you can revert your queue manager to the original version.
- You saved the queue manager customization macros and JCL for running with the Version 7.1 or Version 8.0 target libraries.

You can re-create the customization for Version 7.1 or Version 8.0 , if the originals are not available to you.

About this task

To restart the queue manager, so that it runs at the version where it was migrated from, just requires that you switch back to using the libraries for the previous version.

Note that it is not necessary to roll back the early code for this installation when reverting your queue manager to an earlier version.

Procedure

1. Stop the listener, channel initiator, and queue manager.
2. Switch back to use the MSTR and CHINIT started procedure JCLs with Version 7.x or Version 8.0 libraries.

   In case data set aliases are used for load libraries, switch the aliases to refer to the Version 7.x or Version 8.0 libraries.

   For example, an alias named `MQM.MQP1.SCSLOAD`, referring to `MQM.MQV800.SCSLOAD`, needs to change to refer to `MQM.MQV7xx.SCSLOAD`.
3. Restart the queue manager, using the system parameter module (CSQZPARM) used with IBM WebSphere MQ Version 7.x, or IBM MQ Version 8.0, prior to migration, and linking to the Version 7.x, or Version 8.0, code.

   Until you have verified the startup, start the queue manager, channel initiator, and listener separately, checking for errors on the console after each component is started. If the startup runs cleanly, combine the startup of all three components in the production environment.
   a. Start the queue manager.
   b. Start the channel initiator.
   c. Start the listener.
4. Verify correct functioning of existing applications.

Results

If the queue manager cannot be reverted to the previous release by following the preceding procedure, for example, because it has been started with OPMODE set to (NEWFUNC,900), the queue manager can only be reverted to the previous release by recovering the page sets, BSDSs, and active logs from back up copies taken before the migration to IBM MQ for z/OS Version 9.0.

All updates made since the back up was taken will be lost. See How to back up and recover pagesets for more information on backing up IBM MQ resources.

Post migration tasks: z/OS

Follow the steps to perform the tasks you need to carry out after migrating a single IBM MQ queue manager on z/OS,

About this task

After you have migrated an IBM MQ queue manager on z/OS you need to carry out the detailed steps in this topic, using the links within this overview.

1. Check the changes in behavior made by default configuration changes; see Step 1
2. Modify the backup jobs to refer to the target version of IBM MQ libraries; see Step 2
3. Configure Advanced Message Security; see Step 3
4. Perform a full regression test; see Step 4
5. Update the ZPARM module if you have not already done so; see Step 5
6. Set OPMODE to NEWFUNC; see Step 6 on page 804
7. Exploit the new functions provided by the migrated queue manager; see Step 7 on page 804
8. Consider client application migration; see Step 8 on page 804

.

Procedure

1. Check the changes in behavior made by default configuration changes. The default values of some properties might have been changed in the new version, which can lead to changes in behavior.

   SHARECNV allows multiple connections to the queue manager to permit the use of the same TCP/IP connection. If a client is using Version 6 code to connect to a version 7, or later, IBM MQ queue manager, SHARECNV is set to 0 automatically; see Default behavior for more details about this change.

   On z/OS, you can reverse queue manager migration as long as you have not enabled new function. You enable new function by setting the **`OPMODE`** parameter to `(NEWFUNC,900)` ; see OPMODE for more information.
2. Modify backup, and other administrative, jobs to refer to the target version of IBM MQ libraries, such as backup IBM MQ objects and MAKEDEF jobs. For example using CSQUTIL COMMAND MAKEDEF(..); see Using the COMMAND function of CSQUTIL.

   You should also backup channel authentication records, which were introduced in IBM WebSphere MQ Version 7.1.0.
3. If the queue manager is configured to use Advanced Message Security (AMS) perform the steps in the Post migration tasks for Advanced Message Security section of the Migrating Advanced Message Security topic.
4. Perform a full regression test.
5. Update the system parameter (ZPARM) module if required. Tailor the ZPARM sample to use new IBM MQ libraries and generate new ZPARM.

   There are no changes to the ZPARM between IBM MQ Version 8.0 and IBM MQ Version 9.0.

   Therefore, you do not need to perform this task for a queue manager that has been migrated from Version 8.0.

   Attention: If you re-create the ZPARM for a CD Version 9.0.0 queue manager that has been migrated from Version 8.0, and had previously been running at **`OPMODE`**`=(NEWFUNC,800)`, you must set the value of **`OPMODE`** to `(NEWFUNC,900)` to allow the continued availability of Version 8.0 new functions.

   You must do this, only if you are satisfied with the stability of the latest version, and do not need to revert to the previous version.

6. CD Set OPMODE in ZPARM JCL to NEWFUNC, and recompile the JCL for a Version 9.0.0 Long Term Support (LTS) release queue manager.

   Attention: You do not need to change OPMODE for a Continuous Delivery (CD) release queue manager at this stage.

   For more information about NEWFUNC, see OPMODE.
7. Exploit the new functions provided by the migrated queue manager. Your queue manager has been fully migrated to a new version level, and you can take benefit of new capability now.

   Review What's new in IBM MQ Version 9.0 and check which features best serve your business needs. Plan your action to develop new applications, or changing configurations, to enable those features.
8. Migrate client applications. Client applications can be considered any time throughout the migration phase.

   Clients are backwards and forwards compatible. It is advisable to migrate the client libraries to same level as the queue manager, or later, so that the latest function is available.

Results

You have completed the migration of a single IBM MQ for z/OS queue manager.

Adding a new queue-sharing group to an existing Db2 data sharing group in the latest version on z/OS

z/OS

Follow these steps to add a new queue-sharing group to an existing Db2 data sharing group in the latest version of the product. You must apply the migration and toleration PTFs to queue managers, in the previous version, in any of the queue-sharing groups before adding a queue-sharing group.

Before you begin

1. Review your Db2 data-sharing requirements. A single Db2 data-sharing group can be used to support multiple IBM MQ queue-sharing groups.
2. You can add a new queue-sharing group to a Db2 data-sharing group that already supports IBM MQ queue-sharing groups containing queue managers for the previous version. You must ensure that the migration and toleration PTFs have been applied. The Db2 tables used by IBM MQ must be configured for the latest version queue managers.

About this task

Queue-sharing group migration affects steps 4 on page 777, 8 on page 778, 11 on page 778, and 12 on page 779 of "Reviewing and modifying queue manager customizations from the previous release on z/OS" on page 775

Attention: CD Steps 1 and 2, are necessary only if all the queue managers in the queue-sharing groups, in the Db2 data sharing group, are at IBM WebSphere MQ Version 7.1 or earlier.

Procedure

1. Customize the `CSQ4570T` and `CSQ4571T` samples, in *thlqual*.`SCSQPROC`, supplied with the latest version of the IBM MQ for z/OS product.

   The header information in `CSQ4570T` and `CSQ4571T` describes how to customize the samples.

   Delete or bypass the step that runs `MIGRATE QSG`.
2. Run the customized `CSQ4570T` and `CSQ4571T` jobs.
3. Set up the coupling facility.

   See Task 10: Set up the coupling facility.

4. Customize and include the initialization input sample *thlqual*.SCSQPROC(CSQ4INSS) in the CSQINP2 data set.

   See step 11 on page 778 in "Reviewing and modifying queue manager customizations from the previous release on z/OS" on page 775.
5. Add the IBM MQ entries to the Db2 data-sharing group using the **CSQ5PQSG** program.

   See Task 16: Add the IBM MQ entries to the Db2 data-sharing group.
6. Tailor the system parameter module to add Db2 data-sharing group and IBM MQ queue-sharing group information.

   See step 12 on page 779 in "Reviewing and modifying queue manager customizations from the previous release on z/OS" on page 775.

Queue-sharing group migration

z/OS

You can combine queue managers from different releases in a queue-sharing group. Limit the time you manage a mixed group to only as long as it takes to migrate all the queue managers to the same command level. You cannot combine a queue manager at Version 9.0 or later in the same queue-sharing group as queue managers earlier than Version 7.1. You must update all queue managers in a queue-sharing group with a coexistence PTF, before migrating any of them.

When you migrate queue managers in a queue-sharing group, aim to migrate all the queue managers to the new version as soon as you can. Queue-sharing groups can contain queue managers with a restricted mixture of versions. A mixture of queue managers in a queue-sharing group is supported so that you can migrate and test the upgrade of each queue manager. Migrate each queue manager, one at a time, leaving the queue-sharing group running. Mixed groups are harder to administer, than if all the queue managers are at the same version.

Note: LTS release and CD release queue managers, with the same version and release numbers, can coexist in a queue-sharing group without the need for a coexistence PTF.

Related reference:

"MQSC commands in a mixed queue-sharing group on z/OS" on page 676
Existing **MQSC** commands using new keywords and attribute values can be entered for routing to a migrated queue manager. You can enter the commands on any queue manager. Route the commands using **CMDSCOPE**. Commands with new keywords and attribute values, or new commands, routed to a previous version of queue manager, fail.

"Properties of objects in a mixed queue-sharing group on z/OS" on page 676
Attributes that did not exist in earlier versions can be created and altered on queue managers of a later version in a mixed queue-sharing group. The attributes are not available to queue managers in the group that are at an earlier level.

"Queue-sharing group coexistence on z/OS" on page 676
A queue-sharing group can contain queue managers running on IBM WebSphere MQ Version 7.1.0, and on later releases. The queue managers can access the same shared queues and other shared objects. Queue managers running earlier versions of the product must have the coexistence PTF applied for the latest release.

Migrating queue-sharing groups from a previous version of the product on z/OS: z/OS

You can migrate one or more existing queue-sharing groups containing queue managers in a previous version of a product to the latest version. At no stage is an outage of the entire queue-sharing group required.

Before you begin

1. Read the topic, "Queue-sharing group migration" on page 805, and its related references, especially "Queue-sharing group coexistence on z/OS" on page 676.

About this task

Migrating each queue manager comprises the bulk of the work of migrating a queue-sharing group. Approach migrating a queue-sharing group as requiring some extra tasks that must be performed during the migration of each queue manager. A good approach is to create a migration plan incorporating queue-sharing group migration; see "Planning to migrate to the latest release on z/OS" on page 772.

Queue-sharing group migration affects steps 4 on page 777, 8 on page 778, 11 on page 778, and 12 on page 779 of "Reviewing and modifying queue manager customizations from the previous release on z/OS" on page 775

Procedure

1. Apply IBM MQ for z/OS migration and toleration [11] PTFs for the latest version of the product to the earlier version code; see IBM MQ Support, Migration PTFs.
   a. Apply the PTFs to the libraries of the earlier version of the product. If you need to rebind a plan or package, follow the instructions in the PTFs, and rebind the packages using the instructions in `CSQ45BPK`.

      If you rerun the bind of the plan, or package, while queue managers are active, the bind fails with a locking problem if any queue manager in the DSG using the plan is active.

      You can either shut down the queue managers using the plans, or suspend the queue managers use of Db2. See Suspending a connection to Db2 for further information.
   b. Perform the additional hold action tasks of binding new and changed DBRMs into plans
   c. Stop and restart each queue manager so that it picks up the new code level.
   d. Perform testing of the new code level.

   These steps can be performed at any time in preparation for a migration to the latest version of IBM MQ for z/OS, or as part of normal maintenance. It is not dependent on the latest version being available.

   Migrating an earlier version queue manager to a later version queue manager within a queue-sharing group is restricted. The restrictions are, that all the earlier version queue managers in the queue-sharing group must have been started at the *same* earlier version, and that all the earlier version queue managers have had the "earlier version backwards migration from the latest version, and coexistence with the latest version" PTFs applied.

   After a queue manager for the latest version has been started in a queue-sharing group, starting an earlier version queue manager is restricted. You cannot start an earlier version queue manager as a member of the group, unless it has migration and toleration PTFs applied.

   The latest version requires new Db2 tables, and additional changes to existing Db2 tables. The PTF changes some of the Db2 operations performed by the earlier version queue manager. The changes make an earlier version queue manager compatible with the latest version.

   The PTF contains a new set of Database Request Modules (DBRM). After binding Db2 with these DBRMs, you have two sets of plans: one set for queue managers without the PTFs and the other set for queue managers with the PTFs applied.
2. Migrate your Db2 tables.

   You must have applied the migration and toleration PTFs to all the queue managers in the queue-sharing group before migrating Db2 tables.

11. The "migration and toleration" PTFs are also known as the "backward migration and coexistence" PTFs. They are the same PTFs.

You can either migrate the Db2 tables one queue-sharing groups at a time, or all queue-sharing groups at the same time. For more information, read the header information in the jobs.

- Migrate the tables for all the queue-sharing groups at the same time.

  Attention: CD Steps a and b, are necessary only if all the queue managers in the queue-sharing groups, in the Db2 data sharing group, are at IBM WebSphere MQ Version 7.1 or earlier.

  a. Customize the `CSQ4570T` and `CSQ4571T` samples, in *`thlqual`*`.SCSQPROC`, supplied with the latest version of the IBM MQ for z/OS product.

     The header information in `CSQ4570T` and `CSQ4571T` describes how to customize the samples.

  b. Run the customized `CSQ4570T` and `CSQ4571T` jobs.

  c. Customize the `CSQ45BPL` and `CSQ45GEX` samples, in *`thlqual`*`.SCSQPROC`

     The header information in `CSQ45BPL` and `CSQ45GEX` describes how to customize the samples.

  d. Run the customized jobs, `CSQ45BPL` and `CSQ45GEX`.

     The customized jobs are run as part of step 8 on page 778 of “Reviewing and modifying queue manager customizations from the previous release on z/OS” on page 775.

     This step binds the latest version DBRMs into plans, and grants execute authority to them.

- Migrate the tables and bind the plans.

  Attention: CD Steps a and b, are necessary only if all the queue managers in the queue-sharing groups are at IBM WebSphere MQ Version 7.1 or earlier.

  a. Customize the `CSQ4570T` and `CSQ4571T` samples, in *`thlqual`*`.SCSQPROC`, supplied with the latest version of the IBM MQ for z/OS product.

     The header information in `CSQ4570T` and `CSQ4571T` describes how to customize the samples.

     Edit step 8 on page 778 of “Reviewing and modifying queue manager customizations from the previous release on z/OS” on page 775 that executes the `MIGRATE QSG` function, specifying the name of the first queue-sharing group that is to be migrated.

  b. Run the customized `CSQ4570T` and `CSQ4571T` jobs.

  c. Customize the `CSQ45BPL` and `CSQ45GEX` samples, in *`thlqual`*`.SCSQPROC`

     The header information in `CSQ45BPL` and `CSQ45GEX` describes how to customize the samples.

  d. Run the customized jobs, `CSQ45BPL` and `CSQ45GEX`.

     The customized jobs are run as part of step 8 on page 778 of “Reviewing and modifying queue manager customizations from the previous release on z/OS” on page 775.

     This step binds the latest version DBRMs into plans, and grants execute authority to them.

Migrating a queue manager cluster

You can migrate queue managers in a cluster all at once, or one at a time, which is called a staged migration. Migrate full repository queue managers in a cluster before partial repository queue managers. You must consider what the effect is of migrating some queue managers in a cluster, before all the queue managers are migrated.

Before you begin

Before starting the migration, check that no cluster-specific migration issues are identified for the migration you are intending to perform.

Consider the following issues that relate to migrating a queue manager cluster:
- Minimizing application outages.
- Measuring and verifying migration success and planning for backward migration if there are any migration problems.
- Taking advantage of new IBM MQ features
- Managing the migration of a cluster in the context of the wider IBM MQ network and the systems architecture of your organization.

About this task

Cluster queue managers can participate in clusters with other queue managers running at different versions, which is why a staged migration is possible. Being able to stage a migration is important, as migrating each queue manager in a cluster takes time. By staging the migration, which leaves other queue managers that are in the cluster running, you reduce the effect of queue manager downtime on applications.

Migrate queue managers with full repositories first. Then migrate the other queue managers, which have partial repositories, one at a time. Complete migration of the entire cluster before starting to use new functions.

If you do have to start using new functions before completing migration of the entire cluster, you might need to refresh the partial repositories. After each migration of a queue manager with a partial repository, issue the **REFRESH CLUSTER** command on the newly migrated queue manager. The command updates the cluster records in the newly migrated queue manager, potentially receiving updates for any new attributes. Do not do this step if you migrated the entire cluster before using new function. The **REFRESH CLUSTER** command takes a long time for all the changes to work through the cluster.

Note: 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 Refreshing in a large cluster can affect performance and availability of the cluster.

If full repositories are not migrated before partial repositories, the cluster continues to work, but without all the new features in a version working as expected. To work predictably, the full repository queue managers must be at the new command level to be able to store information from the rest of the cluster that arises from using new features.

For example, the information might be a new channel attribute, such as shared conversations, which were introduced in Version 7.0. Information about the shared conversation attribute of a channel between two other Version 7.0.1 queue managers can be stored in a Version 7.0 full repository, but not in a Version 6.0 repository. If information about a channel with the shared conversation attribute is updated from the

Version 6.0 full repository, the definition loses its shared conversation attribute. "How mixed version cluster repositories are updated" explains how information is updated in a mixed-version cluster.

Notes:

1. In exceptional circumstances, it might be necessary to upgrade some of your partial repositories before your full repositories.

   While the product supports this configuration, in this situation be very careful to avoid use of any new clustering function on the partial repositories, until your full repositories have been upgraded, to avoid unexpected results.
2. If a queue manager is a member of a cluster, and is running at a release earlier than Version 6.0, you must migrate the queue manager to Version 7.0.1, before migrating it to the latest release. You must start the queue manager after the first migration step, before proceeding to Version 9.0.

Procedure

- For information about creating a migration plan for a queue manager cluster, see "Creating a migration plan for a queue manager cluster" on page 810.
- For information about creating a backout plan for the migration of a queue manager cluster, see "Creating a backout plan for queue manager cluster migration" on page 811.
- For information about how to migrate one queue manager in a queue manager cluster, see "Migrating one cluster queue manager" on page 812.

How mixed version cluster repositories are updated

Repositories store records for an object in a cluster in the version of the record format that matches the version of the queue manager hosting the repository. Repository queue managers forward object records, before they are stored, in the format that they are received in. The recipient ignores fields from a newer version, and uses default values for fields that are not present in the record.

Cluster repositories hold records that represent objects, for example, a queue record represents a cluster queue. A full repository holds records for all objects in the cluster. Partial repositories hold records for local objects and remote objects that are used locally. A repository record can hold information only about attributes at the same command level as the queue manager holding that repository. So for example, a Version 8.0 repository contains only Version 8.0 level attribute information. A repository contains all Version 8.0 records, plus Version 9.0 records containing additional Version 9.0 attributes.

A repository stores a record it receives in its own version. If the record it receives is at a later version, the later version attributes are discarded when the record is stored. A Version 8.0 queue manager receiving information about a Version 9.0 queue manager stores only Version 6.0 information. A Version 9.0 repository receiving a Version 8.0 record stores default values for attributes introduced in the version 7. The defaults define the values for the attributes that are not included in the record it receives.

A repository normally sends records in its own version format, which is the same as the format it has stored them in. There is one exception to this rule. When a full repository receives a record from a partial repository, it is immediately forwarded in the same format. So if a Version 8.0 full repository were to receive a record from a Version 9.0 partial repository, it would forward the Version 9.0 record. It sends the record to any other full repositories, and any other partial repositories that have subscriptions that match the record.

A partial repository reflects whichever full repository sent it the latest update to a record. As a consequence, you might see the information held by a Version 9.0 partial repository for new Version 9.0 attributes changing unexpectedly. The values might change from actual Version 9.0 information to default values. The changes occur if the full repositories in the cluster are at different levels. Migrate full repositories first to avoid instability.

A partial repository sends information about its objects to a full repository periodically at least once every 27 days. Information is sent about any object when it is altered or defined. See How long do the queue manager repositories retain information?

After migrating all full repositories to Version 9.0, some attributes might hold default values. The attributes might hold default values in place of actual values, if a repository has not received an update. You can refresh the repository in either of two ways:

- Alter the object which the record containing default values represents, for example, using `ALTER QL` for a local queue. The alteration forces the local repository to send the record again.
- Issue the **`REFRESH CLUSTER`** command on the partial repository which holds the record containing default values. **`REFRESH CLUSTER`** forces the partial repository to discard the record containing default values and get a new record as required.

  Note: 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 Refreshing in a large cluster can affect performance and availability of the cluster.

In summary, for the most predictable, and fastest migration, when you stage cluster migration do these steps in the following order:

1. Migrate the queue managers with full repositories.
2. Migrate the queue managers with partial repositories.
3. Start using new function in the cluster.

Note: In exceptional circumstances, it might be necessary to upgrade some of your partial repositories before your full repositories.

While the product supports this configuration, in this situation be very careful to avoid use of any new clustering function on the partial repositories, until your full repositories have been upgraded, to avoid unexpected results.

Related information:

How long do the queue manager repositories retain information?

Creating a migration plan for a queue manager cluster

Before carrying out the migration of a queue manager cluster, plan what you are going to do. Identify the roles that different queue managers play in the cluster, and decide in what order to migrate the queue managers.

Procedure

- What queue manager and application migration issues must be dealt with between the old and new versions?
- What system architecture and change control procedures must you consider?
- Consider migration questions specific to clusters, such as migrating full repositories first, and migrating overlapping clusters.
- Are any of the queue managers in a queue-sharing group, or part of a high-availability solution?
- Is the cluster a publish/subscribe cluster? Which queue manager is a cluster topic host?
- Decide whether to carry out a staged migration, or migrate all queue managers at the same time.
- Do you have a test system to migrate , and a production system?
- Document and test the plan before migrating production queue managers.

Related concepts:

"Application migration and interoperation" on page 652
IBM MQ supports running applications compiled and linked against previous versions of IBM MQ, with later levels of IBM MQ. IBM MQ applications might require migration between Version 7.1 and the latest version.

"How mixed version cluster repositories are updated" on page 809
Repositories store records for an object in a cluster in the version of the record format that matches the version of the queue manager hosting the repository. Repository queue managers forward object records, before they are stored, in the format that they are received in. The recipient ignores fields from a newer version, and uses default values for fields that are not present in the record.

"Queue manager migration" on page 648
After upgrading an installation, queue manager migration might be required. Migration takes place when you start a queue manager. You can remove an upgrade before you have started a queue manager. However, if you remove the upgrade after a queue manager has been started, the queue manager will not work.

z/OS "Queue-sharing group migration" on page 805
You can combine queue managers from different releases in a queue-sharing group. Limit the time you manage a mixed group to only as long as it takes to migrate all the queue managers to the same command level. You cannot combine a queue manager at Version 9.0 or later in the same queue-sharing group as queue managers earlier than Version 7.1. You must update all queue managers in a queue-sharing group with a coexistence PTF, before migrating any of them.

Related tasks:

"Migrating a queue manager in a high-availability configuration" on page 814
High-availability configurations of queue managers can increase the availability of IBM MQ applications. If a queue manager, or server fails, it is restarted automatically on another server. You can arrange for IBM MQ MQI client applications to automatically reconnect to the queue manager. Server applications can be configured to start when the queue manager starts.

Related information:

Availability of cluster topic host queue managers

Creating a backout plan for queue manager cluster migration

Before performing a migration, decide on a backout plan in case of failure.

Before you begin

What backout capabilities do the queue managers in the cluster support?

z/OS If the libraries of the earlier level of IBM MQ include the appropriate PTFs to be able to backward migrate, and NEWFUNC mode is not enabled at the higher level, queue managers running on z/OS can be reverted to an earlier level by changing the load libraries.

On other platforms, the only backout option is to restore a queue manager to a previous state. In restoring a queue manager, you lose any persistent changes since the queue manager started running at the new level.

About this task

The backout plan must consider how to maintain the availability of the cluster. It must deal with any issues arising from migrating a queue manager in the cluster.

Procedure

The backout plan must describe the following points:

- What constitutes a successful migration.
- The conditions that trigger the backout procedure.
- Alternative backout actions, such as:
  1. Suspending a queue manager from the cluster.
  2. Backward migration
  3. Keeping a queue manager offline until an external issue is resolved.

Related concepts:

"Restoration of a queue manager to an earlier version on all platforms" on page 650
You can remove an upgrade before you have started a queue manager. However, if you remove the upgrade after a queue manager has been started, the queue manager will not work.

Migrating one cluster queue manager

Follow these steps to migrate a single queue manager in a cluster, starting with a queue manager in your test system. Base these steps on your cluster migration plan.

Procedure

1. Suspend the queue manager that you want to migrate from the cluster:
   a. Issue the **MQSC** command:
   ```
   SUSPEND QMGR CLUSTER(cluster name)
   ```
   b. 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.
2. Save a record of all cluster objects known by this queue manager. This data is used after migration to check that objects have been migrated successfully.
   a. Issue the command to view cluster queue managers.
   ```
   DISPLAY CLUSQMGR(*)
   ```
   b. Issue the command to view cluster queues.
   ```
   DISPLAY QC(*)
   ```
   c. Issue the command to view cluster topics.
   ```
   DISPLAY TCLUSTER(*)
   ```
3. Save a record from the full repository of its view of the cluster objects owned by this queue manager. The record is used after migration to check that objects have been migrated successfully.
   a. Issue the command on the full repositories to display this queue manager.
   ```
   DISPLAY CLUSQMGR(migrated queue manager name)
   ```
   b. Issue the command on the full repositories to display the cluster queues for this queue manager
   ```
   DISPLAY QC(*) WHERE(CLUSQMGR EQ migrated queue manager name)
   ```
   c. Issue the command on the full repositories to display the cluster topics for this queue manager.
   ```
   DISPLAY TCLUSTER(*) WHERE(CLUSQMGR EQ migrated queue manager name)
   ```
4. Migrate the queue manager.

   Do one of the queue manager migration tasks, depending on the platform; see "Migrating a queue manager on Windows" on page 691.

   The queue manager migration process is, in outline:

   a. Stop the queue manager.
   b. Take a backup of the queue manager.

   c. Install the new version of IBM MQ.
   d. Restart the queue manager.
5. Ensure that all cluster objects have been migrated successfully.
   a. Issue the command to view cluster queue managers and check the output against the data saved before migration.
      ```
      DISPLAY CLUSQMGR(*)
      ```
   b. Issue the command to view cluster queues and check the output against the data saved before migration.
      ```
      DISPLAY QC(*)
      ```
   c. Issue the command to view cluster topics and check the output against the data saved before migration.
      ```
      DISPLAY TCLUSTER(*)
      ```
6. Check that the queue manager is communicating with the full repositories correctly.
7. Check that cluster channels to full repositories can start.
8. Check that the full repositories still have information about the migrated cluster queue manager, its cluster queues, and its cluster topics.
   a. Issue the command on the full repositories and check the output against the data saved before migration.
      ```
      DISPLAY CLUSQMGR(migrated_queue_manager_name)
      ```
   b. Issue the command on the full repositories and check the output against the data saved before migration.
      ```
      DISPLAY QC(*) WHERE(CLUSQMGR EQ migrated_queue_manager_name)
      ```
   c. Issue the command on the full repositories and check the output against the data saved before migration.
      ```
      DISPLAY TCLUSTER(*) WHERE(CLUSQMGR EQ migrated_queue_manager_name)
      ```
9. Test that applications on other queue managers can put messages to queues owned by the migrated cluster queue manager.
10. Test that applications on the migrated queue manager can put messages to the queues owned by other cluster queue managers.
11. Resume the queue manager by issuing the following command:
    ```
    RESUME QMGR CLUSTER(cluster name)
    ```
12. Closely monitor the queue manager and applications in the cluster for a while.

What to do next

When you have completed the migration of one queue manager in a cluster, on your test system, complete the migration of the other queue managers in each cluster on the test system.

When you have competed the migration of all of the queue managers on your test system, migrate each of the queue managers on your production system.

Related concepts:

"Queue manager migration" on page 648
After upgrading an installation, queue manager migration might be required. Migration takes place when you start a queue manager. You can remove an upgrade before you have started a queue manager. However, if you remove the upgrade after a queue manager has been started, the queue manager will not work.

Related information:

DISPLAY CLUSQMGR

DISPLAY QUEUE

RESUME QMGR

SUSPEND QMGR

Migrating a queue manager in a high-availability configuration

High-availability configurations of queue managers can increase the availability of IBM MQ applications. If a queue manager, or server fails, it is restarted automatically on another server. You can arrange for IBM MQ MQI client applications to automatically reconnect to the queue manager. Server applications can be configured to start when the queue manager starts.

About this task

Multi For IBM MQ for Multiplatforms, high-availability configurations are implemented either by using a high-availability cluster solution or by using multi-instance queue managers. Red Hat Cluster Suite or Microsoft Cluster Service (MSCS) are examples of high-availability cluster solutions.

z/OS For IBM MQ for z/OS there are several alternative techniques to increase queue manager availability; see Availability on z/OS. Migration considerations on z/OS depend on the availability techniques that are employed, and are not described in this topic. The term high-availability configuration refers only to queue managers in configurations on platforms other than z/OS.

The overall principles involved in queue manager migration in a high availability configuration are the same, whether you are migrating a multi-instance queue manager or a high-availability cluster. In either case, the principles are as follows:

1. You must not restart a queue manager at a lower command level than the one it was previously running.
2. You cannot upgrade the code an active queue manager is running.
3. You cannot back up an active queue manager.

Procedure

- To migrate a multi-instance queue manager, see "Migrating a multi-instance queue manager" on page 815.
- To migrate a high availability cluster queue manager, see "Migrating a high-availability cluster queue manager" on page 816.

Related tasks:

"Migrating an MSCS configuration on Windows" on page 717
Migrate queue managers in a Microsoft Cluster Service (MSCS) configuration one node at a time, following these instructions.

Migrating a multi-instance queue manager

Follow the steps listed to migrate a queue manager in a multi-instance queue manager configuration.

Before you begin

The following terms are relevant:

active queue manager instance
: A queue manager instance that has been started permitting standby instances, and is running.

standby queue manager instance
: A queue manager instance that has been started permitting standby instances, and is in standby. It is ready to take over from the active instance automatically.

Procedure

Base your migration procedure on the following steps:

1. Before you start the migration process, create a different queue manager on a server, on which you have installed the upgrade.
2. Test the upgrade by performing whatever verification checks that your organization requires.
3. If you have a pool of servers that you pick from, when starting a queue manager instance, upgrade IBM MQ on the servers that are in the pool and are neither active or acting as a standby.
4. Stop the standby queue manager instance. Ensure that you have no system management procedure running that restarts the instance automatically.
5. If you do not have a pool of servers, upgrade IBM MQ on the server that was running the standby instance
6. Decide whether downtime or recoverability is more important in the migration.
7. Optional: Follow this procedure if recoverability is more important, and you must take a backup:
   a. Stop the active queue manager instance, without switching to any standby.
   b. Back up the queue manager
   c. Start a queue manager instance, permitting standbys, on one of the upgraded servers.
   d. If you have a pool of upgraded servers, start another one, permitting standbys.
8. Optional: Follow this procedure if availability is more important. You do not need to take a backup.
   a. Start a queue manager instance as a standby on one of the upgraded servers.
   b. Stop the active queue manager instance, switching to the standby.
   c. If you have a pool of upgraded servers, start another one, permitting standbys.
9. Upgrade the IBM MQ code on the server that was the active queue manager instance.
10. Start the server as the standby instance if you have not already started a standby.

"Migrating a queue manager in a high-availability configuration" on page 814
High-availability configurations of queue managers can increase the availability of IBM MQ applications. If a queue manager, or server fails, it is restarted automatically on another server. You can arrange for IBM MQ MQI client applications to automatically reconnect to the queue manager. Server applications can be configured to start when the queue manager starts.

"Migrating a high-availability cluster queue manager"
Follow the steps listed to migrate a queue manager in a high-availability queue manager configuration.

Migrating a high-availability cluster queue manager

Follow the steps listed to migrate a queue manager in a high-availability queue manager configuration.

Before you begin

The following terms are relevant:

active server
The running server or active queue manager instance

passive server
A server that is ready to take over from the active server automatically.

inactive server
A server that is not prepared to take over automatically. The server might have been removed from the cluster, or taken offline in some way.

Procedure

Base your migration procedure on the following steps. The details depend on the specific commands in the cluster concerned.

1. Before you start the migration process, create a different queue manager on a server on which you have installed the upgrade.
2. Test the upgrade by performing whatever verification checks that your enterprise requires.
3. Form two cluster pairs if you have four servers available. With two pairs, the queue manager can continue to run in a cluster-pair at the old command level. When you are ready, you can transfer the queue manager to the pair of servers at the new command level.
4. Remove a passive server from the cluster. Ensure that the cluster cannot automatically restart the server. The server is made inactive.
5. Create a second location for the upgraded code, if a high-availability cluster is using a common location for IBM MQ code.
6. Install, or upgrade, IBM MQ code using the server that is not now running the queue manager.
7. Verify the upgrade by creating a different queue manager on the server, and performing whatever verification checks that your organization requires.
8. If more than half the servers remain in the cluster, remove a server, upgrade IBM MQ, and verify the upgrade. Each server is made inactive as part of the process. Continue until half the servers are upgraded.
9. If your active server is part of a remaining cluster, deactivate the passive servers so that the cluster cannot reactivate them automatically.
10. Decide whether downtime or recoverability is more important in the migration.
11. Optional: Follow this procedure if recoverability is more important:
    a. Stop the queue manager and remove the server from the cluster.
    b. Back up the queue manager.
12. Optional: Follow this procedure if downtime is more important:
    a. Add the migrated servers back into the cluster, as passive servers.

   b. Switch the remaining server in the high-availability server cluster over to one of the passive servers. The switch causes the running queue manager to stop, and restarts it on one of the passive servers.
13. Upgrade any remaining high-availability servers, and add them back into the cluster.

“Migrating a queue manager in a high-availability configuration” on page 814
High-availability configurations of queue managers can increase the availability of IBM MQ applications. If a queue manager, or server fails, it is restarted automatically on another server. You can arrange for IBM MQ MQI client applications to automatically reconnect to the queue manager. Server applications can be configured to start when the queue manager starts.

“Migrating a multi-instance queue manager” on page 815
Follow the steps listed to migrate a queue manager in a multi-instance queue manager configuration.

Migrating replicated data queue managers

Linux MQ Adv.

When you need to migrate replicated data queue managers (RDQMs), you must upgrade all nodes in a sequence. Do not try to operate with the nodes at different levels.

About this task

The upgrade sequence for HA RDQM configurations consists of suspending a node, uninstalling IBM MQ and RDQM support, installing the newer version of IBM MQ, then resuming the node. You then move on and repeat this sequence on the next node. Following this sequence ensures that your queue managers continue to run on one of the nodes in the HA group while the migration is in progress.

V 9.0.5 The upgrade sequence for DR RDQM configurations consists of upgrading the recovery node, running the DR queue managers on the newly upgraded recovery node, upgrading the primary node, switching the DR queue managers back to running on the primary node.

Scripts are provided that you run to uninstall and install IBM MQ and RDQM.

Related information:
Installing RDQM (replicated data queue managers)

Migrating HA RDQMs

V 9.0.4

Follow this sequence of steps to upgrade all the RDQM nodes in an HA group and so migrate the replicated data queue managers (RDQMs).

About this task

You should upgrade all the nodes in an HA group in the same sequence to avoid operating with the nodes in the group at different levels.

If you have configured your HA group such that one node acts as a primary for all RDQMs, with the other two nodes as secondaries, you should upgrade the secondary nodes first and leave the primary node until last.

The sequence in which you upgrade, and the nodes that are marked as preferred and second preferred locations for RDQMs, affect where the RDQMs fail over to as you upgrade. During the migration sequence, while nodes are running different levels, the options for failing over are limited. An RDQM running on a lower level node can fail over to a higher level node but, once a queue manager has been started at the new level, it cannot fail over to a lower level node. You should choose an upgrade sequence and use preferred and second preferred locations settings to keep queue managers running on the lower level nodes for as long as possible. You should make changes to preferred and second preferred location settings before you suspend nodes, to ensure that the changes are effective immediately.

V 9.0.5 If you are also running DR RDQMs on any of the nodes, you should deal with these queue managers at the same time by following the instructions in "Migrating DR RDQMs."

Procedure

1. To upgrade the first node in the HA group:
   a. Suspend the node from the HA group by running the following command:
      ```
      rdqmadm -s
      ```
      Any RDQMs currently running on the node move to another node in the HA group (their second preference, if one is defined for that RDQM).
   b. Run the uninstall script to uninstall IBM MQ and RDQM.
      ```
      MQ_INSTALLATION_PATH/Advanced/RDQM/uninstallRDQMsupport
      ```
   c. Accept the license for the new version by running the **mqlicense** script.
   d. Run the installation script from the /Advanced/RDQM directory of your installation media, accepting the license when prompted:
      ```
      installRDQMsupport
      ```
   e. If required, set this installation as the primary IBM MQ installation, using the **setmqinst** command. See setmqinst (set IBM MQ installation).
   f. Resume the node in the HA group by entering the following command:
      ```
      rdqmadm -r
      ```
      Any RDQMs that have this node as their preferred location will resume running on this node.
2. Repeat the steps for the second node in the HA group.
3. Repeat the steps for the third node in the HA group.

Related information:

rdqmadm (administer replicated data queue manager cluster)

Migrating DR RDQMs

Linux V 9.0.5

Follow this sequence of steps to upgrade the primary and recovery nodes in a disaster recover replicated data queue manager (DR RDQM) configuration.

About this task

The suggested sequence for upgrading your nodes is to upgrade your recovery node, then run your DR queue managers there while you then upgrade your primary node. When both nodes are upgraded you can restore the original primary and recovery roles.

If you do not require to run your DR queue managers during the upgrade procedure, then you can omit the steps for failing over to the recovery node. You can just stop your DR queue managers and restart them after you have upgraded both nodes.

If you are also running HA RDQMs on either of the nodes, you should deal with these queue managers at the same time by following the instructions in "Migrating HA RDQMs" on page 817.

Procedure

- To upgrade while continuing to run your DR queue managers:
  1. Upgrade your recovery node:
     a. Run the uninstall script to uninstall IBM MQ and RDQM.
        ```
        MQ_INSTALLATION_PATH/Advanced/RDQM/uninstallRDQMsupport
        ```
     b. Accept the license for the new version by running the **mqlicense** script.
     c. Run the installation script from the `/Advanced/RDQM` directory of your installation media:
        ```
        installRDQMsupport
        ```
     d. If required, set this installation as the primary IBM MQ installation, using the **setmqinst** command. See setmqinst (set IBM MQ installation).
  2. Turn the DR queue managers into secondary instances on your primary node by entering the following commands for each queue manager:
     a. Stop the queue manager:
        ```
        endmqm -r QMname
        ```
     b. Make the queue manager into a secondary instance:
        ```
        rdqmdr -m QMname -s
        ```
  3. Run the queue managers on the recovery node by completing the following steps:
     a. Make each queue manager into a primary instance:
        ```
        rdqmdr -m QMname -p
        ```
     b. Start each queue manager:
        ```
        strmqm qmname
        ```
  4. Upgrade the primary node:
     a. Run the uninstall script to uninstall IBM MQ and RDQM.
        ```
        MQ_INSTALLATION_PATH/Advanced/RDQM/uninstallRDQMsupport
        ```
     b. Accept the license for the new version by running the **mqlicense** script.
     c. Run the installation script from the `/Advanced/RDQM` directory of your installation media:
        ```
        installRDQMsupport
        ```
     d. If required, set this installation as the primary IBM MQ installation, using the **setmqinst** command. See setmqinst (set IBM MQ installation).
  5. On the recovery node, make the queue managers into secondary instances once more:
     ```
     rdqmdr -m QMname -s
     ```
  6. On the primary node, make the queue managers into primary instances and start them:
     ```
     rdqmdr -m QMname -p
     strmqm qmname
     ```
- To upgrade while not running your DR queue managers:
  1. Upgrade your recovery node:
     a. Run the uninstall script to uninstall IBM MQ and RDQM.
        ```
        MQ_INSTALLATION_PATH/Advanced/RDQM/uninstallRDQMsupport
        ```
     b. Accept the license for the new version by running the **mqlicense** script.
     c. Run the installation script from the `/Advanced/RDQM` directory of your installation media:
        ```
        installRDQMsupport
        ```
     d. If required, set this installation as the primary IBM MQ installation, using the **setmqinst** command. See setmqinst (set IBM MQ installation).
  2. Stop each queue manager on your primary node:
     ```
     endmqm QMname
     ```
  3. Upgrade the primary node:
     a. Run the uninstall script to uninstall IBM MQ and RDQM.

```
MQ_INSTALLATION_PATH/Advanced/RDQM/uninstallRDQMsupport
```

b. Accept the license for the new version by running the **mqlicense** script.
c. Run the installation script from the /Advanced/RDQM directory of your installation media:

```
installRDQMsupport
```

d. If required, set this installation as the primary IBM MQ installation, using the **setmqinst** command. See setmqinst (set IBM MQ installation).

4. Start the queue managers on your primary node:

```
strmqm qmname
```

Migrating logs on UNIX, Linux, and Windows

ULW

From IBM MQ Version 9.0.4 you can migrate a circular log to a linear log, or from a linear log to a circular log.

Before you begin

Decide whether you want to use linear or circular logging by reviewing Types of logging.

Procedure

- To migrate your queue manager log from being linear to circular, see “Migrating the log of your queue manager from linear to circular.”
- To migrate your queue manager log from being circular to linear, see “Migrating the log of your queue manager from circular to linear” on page 821.

Related concepts:
“Migrating logs to an Advanced Format disk on Windows” on page 719
An Advanced Format disk is one that has 4096 bytes per sector. The following is applicable only to the Windows platform as Advanced Format disks can be used on other platforms, without carrying out a migration procedure.

Migrating the log of your queue manager from linear to circular

ULW V 9.0.4

Follow this sequence of steps to migrate your queue manager log from being linear to circular.

Before you begin

Before you migrate, take a backup of your queue manager.

Decide whether you want to migrate your log in place, or migrate the log to a new location by specifying the **-ld** option on the **migmqlog** command. 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.

Make sure that you have enough space to migrate your log, as the log might grow during migration.

Note the following:
- You do not need to migrate the log of your queue manager in order to use IBM MQ Version 9.0.4.
- **migmqlog** might take some minutes to complete if your log is very large. However, the command outputs progress messages from time to time.
- If, for any reason (for instance due to a power outage) **migmqlog** stops before it has completed the process, rerun the same **migmqlog** command on the partially migrated logs to complete the migration.

- If you specified the **-ld** option, **migmqlog** updates the log path in the `qm.ini` file for you, so when you start your queue manager it will use the migrated log.
- Do not pass a relative path when using the **-ld** option; only use an absolute path.
- **migmqlog** does not update any queue or queue manager objects.

Procedure

1. Login as a member of the `mqm` group.
2. If you have not already done so, take a backup of your queue manager.
3. Run the following command:
   ```
   migmqlog -m QMgrName -lc
   ```
   See **migmqlog** for more information.

Related tasks:
"Migrating the log of your queue manager from circular to linear"
Follow this sequence of steps to migrate your queue manager log from being circular to linear.

Migrating the log of your queue manager from circular to linear

ULW V 9.0.4

Follow this sequence of steps to migrate your queue manager log from being circular to linear.

Before you begin

Before you migrate, take a backup of your queue manager.

Decide whether you want to migrate your log in place, or migrate the log to a new location by specifying the **-ld** option on the **migmqlog** command. 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.

Make sure that you have enough space to migrate your log, as the log might grow during migration.

Note the following:
- You do not need to migrate the log of your queue manager in order to use IBM MQ Version 9.0.4.
- **migmqlog** might take some minutes to complete if your log is very large. However, the command outputs progress messages from time to time.
- If, for any reason (for instance due to a power outage) **migmqlog** stops before it has completed the process, rerun the same **migmqlog** command on the partially migrated logs to complete the migration.
- If you specified the **-ld** option, **migmqlog** updates the log path in the `qm.ini` file for you, so when you start your queue manager it will use the migrated log.
- Do not pass a relative path when using the **-ld** option; only use an absolute path.
- **migmqlog** does not update any queue or queue manager objects.

About this task

Attention: After you have migrated, 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. If you have not already done so, take a backup of your queue manager.
3. Run the following command:
   ```
   migmqlog -m QMgrName -ll
   ```
   See **migmqlog** for more information.
4. Start the queue manager, and set the appropriate image recovery and queue attributes for your environment.
5. Consider when to record manual images for objects that are recoverable.

Related tasks:

"Migrating the log of your queue manager from linear to circular" on page 820
Follow this sequence of steps to migrate your queue manager log from being linear to circular.

Internet Protocol Version 6 (IPv6) migration

This section deals with using IPv4 and IPv6 when you are thinking of installing IBM MQ

General Introduction

The Internet Protocol Version 6 (IPv6) is designed by the Internet Engineering Task Force (IETF) to replace the current version Internet Protocol, Version 4 (IPv4). IPv4 has been around for over 20 years and is one of the primary methods for machines to communicate to each other over the internet. IPv4 is limited to 32-bit addressing for internet addresses. These addresses are needed by all new machines added to the internet and they are beginning to run out. The IETF is the controlling standards body for the Internet and to meet the growing demand for internet addresses has increased the number of digits used for Internet addresses from 32 to 128 bits. IPv6 offers a far larger number (2^{128}) of internet addresses and should solve the address shortage for the foreseeable future. IPv6 is expected to gradually replace IPv4, with the two protocols coexisting for a number of years while this transition period exists. IPv6 also simplifies header formats and improves support for extensions and options, flow labeling capability, and consolidated authentication and privacy capabilities

IBM MQ has the ability for queue managers to communicate using the IPv6 protocol in addition to the existing, IPv4, protocol.

Further information on IPv6 can be found at IPv6 and IPv6 Forum.

IBM MQ platforms that support IPv6

This section lists the IBM MQ platforms that support IPv6.

IPv6 is supported on the following IBM MQ platforms:

- IBM MQ for AIX
- IBM MQ for Linux
- IBM MQ for Sun Solaris
- IBM MQ for HP-UX
- IBM MQ for Windows
- IBM MQ for IBM i
- IBM MQ for z/OS

Key points in migrating to IPv6 and using IBM MQ

This section lists some key points to be aware of when you are thinking of installing IBM MQ and using IPv6.

- IBM MQ recognizes IPv6 hexadecimal addresses (for example fe80:43e4:0204:acff:fe97:2c34:fde0:3485) as well as IPv4 dotted decimal addresses (for example 9.20.9.30).
- For a system running both IPv4 and IPv6 system, the connection name (CONNAME) you specify for a given channel determines the IP protocol for the channel making a connection.

Considerations when implementing IPv6 in a network

This section lists some things that you should consider when you are thinking of installing IBM MQ on an IPv6 network.

- To ensure consistency across the network, you should plan the introduction of IPv6 for the whole network, especially where clusters are involved. For example, although a queue manager is now IPv6 capable, this doesn't imply that the queue managers it can communicate with are also IPv6 capable.
- When setting the domain name server (DNS) or equivalent, consider whether the system on which the target queue manager is running can resolve to an IPv4 address, an IPv6 address or a dual IPv4 and IPv6 address.
- If the system that you are installing IBM MQ on does not support IPv6, IBM MQ will only be able to connect using IPv4.
- For a queue manager running on an IPv6 enabled system to be able to communicate with a queue manager running on an IPv4 enabled system, the IPv4 enabled system must have a host name that resolves to an IPv4 address only.
- If there are multiple domain name servers in an IBM MQ network, each host name used in a channel definition must resolve to the same address (or addresses), regardless of which DNS is used.

Migrating a queue manager to IPv6

This section deals with migrating a queue manager when you are thinking of installing IBM MQ on an IPv6 network.

The IPv6 protocol can only be used by IBM WebSphere MQ Version 6.0 or later. In order to make use of the IPv6 protocol, IBM MQ must be installed on a system that is IPv6 capable.

The preferred IP version that two systems use for communicating (if both IPv4 and IPv6 are available) is determined by a new queue manager attribute IPADDRV. This parameter only has an effect if the host name resolves ambiguously to both an IPv4 address and an IPv6 address.

To migrate a queue manager to use the IPv6 protocol:

1. Configure dual IPv4 and IPv6 protocols on the system where the queue manager to be migrated resides.
2. Install IBM MQ.
3. Add an entry to the DNS to resolve the host name of the system that is to be migrated, to both an IPv4 address and an IPv6 address.
4. Set the IPADDRV parameter to IPv6 (or set the LOCLADDR parameter to resolve to an IPv6 address).

   CAUTION:
   Not all IPv6 software can interpret an IPv4 mapped IPv6 address. If the combination of CONNAME and LOCLADDR results in an IPv4 mapped IPv6 address, ensure that the system hosting the target queue manager is capable of handling this.

   Using mapped addresses can require protocol translators in the IP network.

Migration scenarios (non-cluster topology)

It is possible to come up with a number of different interconnection possibilities, and the following sections aim to help you understand how IBM MQ will work in each case.

Non-cluster migration scenario 1

Three systems exist that are IPv4 only capable. Each system hosts a queue manager (QM1, QM2, and QM3) and each queue manager connects to the other two. All CONNAMEs in the cluster channel definitions are made using DNS names rather than IP addresses.

Enable QM1 to be able to use channels running over IPv6 as follows

1. Upgrade the host system to have dual IPv4 and IPv6 stacks.

   Important: A listener is required for each IP stack.
2. Install the latest version of IBM MQ.
3. Update the DNS table so that it has two entries for the system running QM1; one entry for its IPv4 address and one for its IPv6 address. This enables a DNS name request to return both IPv4 and IPv6 addresses for this host.
4. Set the queue manager IPADDRV attribute to IPv6.

Note: Even with these changes made to support IPv6 addressing, QM1 will still be able to communicate with queue managers (both existing and new ones) that are only IPv4 capable.

Enable QM2 to be able to use channels running over IPv6 as for QM1 above.

- Communications between QM1 and QM2 will now be over IPv6.
- Communications between QM1 and QM3 will still be over IPv4.
- Communications between QM2 and QM3 will still be over IPv4.

With the queue manager IPADDRV attribute set to IPv6, the preference has been set for the queue manager to connect using the IPv6 protocol. If a channel from QM1 to QM3 has LOCLADDR set to a host name which resolves to an IPv6 address, or both IPv4 and IPv6 addresses (with the IPADDRV attribute set to IPv6, the IPv6 address will be returned as that is the preference), this channel will attempt to use the IPv6 protocol. If the IPv6 protocol installed on the QM1 host system is capable of using a mapped address then QM1 will communicate with QM3 over IPv6. Otherwise, the channel will fail to resolve CONNAME.

While QM3 remains a queue manager on an earlier version of the product, you will need to check that all CONNAMEs used to start a channel to QM3 do not resolve to an IPv6 address or dual IPv4 and IPv6 addresses where the IPv6 address could be returned. This would cause QM1 to attempt to start the channel over IPv6 which would fail, as it would be unable to resolve the CONNAME.

It is possible to upgrade a system to have dual IPv4 and IPv6 capability and still run a queue manager on an earlier version of the product, on the system. While it is not recommended to run this type of configuration, as long as the addresses that are returned to this level of queue manager are either IPv4 or an IPv4 mapped version of an IPv6 address, this should work.

Non-cluster migration scenario 2

Three systems exist that are IPv4 only capable. Each system hosts a queue manager (QM1, QM2, and QM3) and each queue manager connects to the other two. All CONNAMEs in the cluster channel definitions are made using IP addresses.

Because addresses have been specified instead of DNS names, to allow a queue manager to connect to another using the IPv6 protocol you will need to duplicate the definitions that use IPv4 addresses between them and provide them with IPv6 addresses instead. The original definitions that use IPv4 addresses will continue to work, but if you intend to take advantage of the IPv6 protocol, you will need to connect using the new definitions.

Enable QM1 to be able to use channels running over IPv6 as follows

1. Upgrade the host system to have dual IPv4 and IPv6 stacks.

   Important: A listener is required for each IP stack.
2. Install IBM MQ.
3. Duplicate the channel, transmission queue and, where applicable, any process definitions using IPv6 addresses where required.

Note: Even with these changes made to support IPv6 addressing, QM1 will still be able to communicate with existing queue managers that are only IPv4 capable.

Enable QM2 to be able to use channels running over IPv6 as for QM1 above.

1. Upgrade the host system to have dual IPv4 and IPv6 stacks.

   Important: A listener is required for each IP stack.
2. Install IBM MQ.
3. Where necessary amend applications to write to the new remote queue (created above for QM1 with the IPv6 addresses).
4. Verify the channels can be started.

The queue managers can now connect as follows:

- QM1 can now connect with QM2 over either IPv4 or IPv6 depending on the channel the application writes its messages to.
- QM1 still connects with QM3 over IPv4 using the original definitions.

Migrating a cluster to IPv6

This section deals with migrating clusters when you are thinking of installing IBM MQ on an IPv6 capable network.

The following gives an overview of approaches that can be taken when migrating a cluster to the latest version of IBM MQ. Due to the variations that can occur within a cluster, the detail is deliberately general and should only be seen as a guide to the likely course of action you will need to take.

Migration scenarios (cluster topology)

Where an IPv6 capable system is to be added to an IBM MQ cluster, all full repository systems in that cluster must be IPv6 capable.

The following scenarios are seen as the ones most likely to occur in customer installations. They describe the changes that are likely to be required.

Scenario 1

A cluster from an earlier version of the product is installed on IPv4 only capable, systems and you need to connect an IPv6 only capable system into the cluster. All CONNAMEs in cluster channel definitions are made using DNS names rather than IP addresses.

When adding a new IPv6 only system to the cluster, identify those queue managers that your new system will communicate with. These include:

- The queue managers your new system will send messages to.
- The queue managers your new system will receive messages from.
- The full repository queue managers

The systems that you have identified must be upgraded before introducing the new system.

Recommended migration procedure:

- Upgrade each of the systems hosting a full repository queue manager as shown in "Migrating a queue manager to IPv6" non-cluster scenario 1.
- Upgrade the remaining cluster systems which need to be IPv6 capable as shown in "Migrating a queue manager to IPv6" non-cluster scenario 1.

With this configuration:

- The new IPv6 only capable system will communicate with the cluster using IPv6 addressing
- All other IPv4 systems that connect into the cluster will continue to communicate using IPv4 addressing
- The systems in the cluster will be able to connect to each other using either IPv4 or IPv6 addressing. The decision as to which address is used depends on whether you have set IPADDRV to specify IPv4 or IPv6 connections.

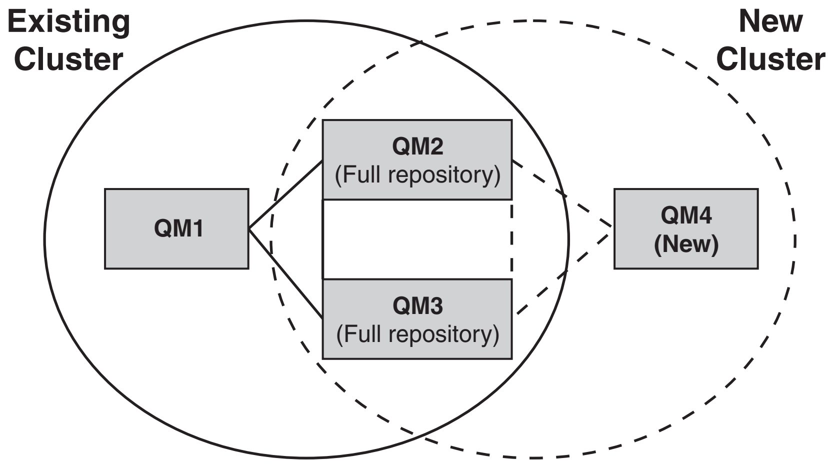
Scenario 2

A cluster from an earlier version of the product is installed on IPv4 only capable systems and you need to connect an IPv6 only capable system into the cluster. Your network does not support adding both IPv6 and IPv4 addresses using the same host name or you are using IP addresses rather than DNS names in the cluster channel CONNAMEs.

The problem here is likely to be that all of the systems cannot be switched to IPv6 simultaneously and some at least must remain only IPv4 capable. The systems that your new IPv6 only system communicates with must be IPv4 and IPv6 capable. We do not recommend simply adding a new set of IPv6 channels into the cluster for the IPv6 system to use, as the IPv4 system would also try to use them, resulting in communication errors.

The recommended approach is:

- Define a new cluster which contains the IPv6 only capable system or systems with new IPv6 addresses and channel definitions. The existing cluster remains, and contains the IPv4 only system definitions. The image below gives a pictorial representation of this. QM1, QM2, and QM3 represent the original IPv4 cluster. QM2, QM3, and QM4 represent the new cluster created to allow the IPv6 only capable system (QM4) to connect into your configuration.
- If you are using DNS names, you can give each of the systems separate DNS names for IPv4 and IPv6 (for example system1_IPv4.ibm.com and system1_IPv6.ibm.com).
- Define a new CLUSRCVR channel and any corresponding CLUSSDR channels using the new IPv6 names or IP addresses on each system in the new cluster. In this way the systems with only IPv4 or IPv6 capability do not see channels which they are not able to use and no communications error will result.



Note: There are both IPv4 and IPv6 definitions connecting the full repositories so that definitions for both new and existing cluster definitions are replicated between them. Also be aware that the queue managers QM1 and QM4 cannot communicate directly because they do not share a common network. They could communicate indirectly, for example by using ALIAS queues defined in the queue managers QM2 and QM3. In the configuration shown above you would need to pay attention to the ordering of application messages flowing between QM2 and QM3 because multiple routes exist, if this is relevant you could use BIND_OPEN to fix the route.

Abbreviated migration scenarios

This section gives some abbreviated scenarios for when you are thinking of installing clusters on IBM MQ

Abbreviated scenarios: Effects of CONNAME and LOCLADDR settings

The following table provides an overview of what will occur for the different TCP/IP stacks (IPv4 only, IPv6 only and dual IPv4 and IPv6 stacks) and given the settings for CONNAME and LOCLADDR the expected connection result.

Note: Using mapped addresses can require protocol translators in the IP network.

Table 96. Effects of CONNAME and LOCLADDR settings

Stack Type	CONNAME setting	LOCLADDR setting	Connection result
IPv4 only stack	IPv4 address		Channel binds to IPv4 stack
	IPv6 address		Channel fails to resolve CONNAME
	Host name resolves to both IPv4 and IPv6 addresses		Channel binds to IPv4 stack
	IPv4 address	IPv4 address	Channel binds to IPv4 stack
	IPv6 address	IPv4 address	Channel fails to resolve CONNAME
	Host name resolves to both IPv4 and IPv6 addresses	IPv4 address	Channel binds to IPv4 stack
	Any address	IPv6 address	Channel fails to resolve LOCLADDR
	IPv4 address	Host name resolves to both IPv4 and IPv6 addresses	Channel binds to IPv4 stack
	IPv6 address	Host name resolves to both IPv4 and IPv6 addresses	Channel fails to resolve CONNAME
	Host name resolves to both IPv4 and IPv6 addresses	Host name resolves to both IPv4 and IPv6 addresses	Channel binds to IPv4 stack
Dual IPv4 and IPv6 stack	IPv4 address		Channel binds to IPv4 stack
	IPv6 address		Channel binds to IPv6 stack
	Host name resolves to both IPv4 and IPv6 addresses		Channel binds to stack determined by IPADDRV
	IPv4 address	IPv4 address	Channel binds to IPv4 stack
	IPv6 address	IPv4 address	Channel fails to resolve CONNAME
	Host name resolves to both IPv4 and IPv6 addresses	IPv4 address	Channel binds to IPv4 stack
	IPv4 address	IPv6 address	Maps an IPv4 CONNAME to an IPv4 mapped IPv6 address. IPv6 implementations that do not support IPv4 mapped IPv6 addressing fail to resolve CONNAME
	IPv6 address	IPv6 address	Channel binds to IPv6 stack
	Host name resolves to both IPv4 and IPv6 addresses	IPv6 address	Channel binds to IPv6 stack
	IPv4 address	Host name resolves to both IPv4 and IPv6 addresses	Maps an IPv4 CONNAME to an IPv4 mapped IPv6 address. IPv6 implementations that do not support IPv4 mapped IPv6 addressing fail to resolve CONNAME
	IPv6 address	Host name resolves to both IPv4 and IPv6 addresses	Channel binds to IPv6 stack
	Host name resolves to both IPv4 and IPv6 addresses	Host name resolves to both IPv4 and IPv6 addresses	Channel binds to IPv6 stack

Table 96. Effects of CONNAME and LOCLADDR settings (continued)

Stack Type	CONNAME setting	LOCLADDR setting	Connection result
IPv6 only stack	IPv4 address		Maps an IPv4 CONNAME to an IPv4 mapped IPv6 address. IPv6 implementations that do not support IPv4 mapped IPv6 addressing fail to resolve CONNAME
	IPv6 address		Channel binds to IPv6 stack
	Host name resolves to both IPv4 and IPv6 addresses		Channel binds to IPv6 stack
	Any address	IPv4 address	Channel fails to resolve LOCLADDR
	IPv4 address	IPv6 address	Maps an IPv4 CONNAME to an IPv4 mapped IPv6 address. IPv6 implementations that do not support IPv4 mapped IPv6 addressing fail to resolve CONNAME
	IPv6 address	IPv6 address	Channel binds to IPv6 stack
	Host name resolves to both IPv4 and IPv6 addresses	IPv6 address	Channel binds to IPv6 stack
	IPv4 address	Host name resolves to both IPv4 and IPv6 addresses	Maps an IPv4 CONNAME to an IPv4 mapped IPv6 address. IPv6 implementations that do not support IPv4 mapped IPv6 addressing fail to resolve CONNAME
	IPv6 address	Host name resolves to both IPv4 and IPv6 addresses	Channel binds to IPv6 stack
	Host name resolves to both IPv4 and IPv6 addresses	Host name resolves to both IPv4 and IPv6 addresses	Channel binds to IPv6 stack

Abbreviated scenarios: System configurations

Table 98 on page 830 gives a number of abbreviated scenarios based on the configuration of the installed queue managers and the IP configuration they are running on. The list is not intended to be exhaustive, but to give a number of examples of what to expect based on the configurations shown.

The abbreviations are combined in Table 98 on page 830 to give the configuration of the systems involved in trying to establish communication. For example:

- v71 + IPv6: Represents a queue manager from an earlier version of the product on a system with a TCP/IP Version 6 stack
- v8 + Dual: Represents a queue manager from the latest version of the product on system with a dual TCP/IP version 4 and Version 6 stack

Table 97. Abbreviations used in system configurations

Abbreviation	Meaning
v71	queue manager from an earlier version of the product
v8	queue manager from the latest version of the product
IPv4	a system using an IPv4 only stack
IPv6	a system using an IPv6 only stack
Dual	a system using both an IPv4 and an IPv6 stack
IPv4DNS	DNS returns an IPv4 address only for host name of system holding the responding queue manager
IPv6DNS	DNS returns an IPv6 address only for host name of system holding the responding queue manager
DualDNS	DNS returns an IPv4 and IPv6 address for host name of system holding the responding queue manager
LOCLADDR4	The LOCLADDR parameter is set to IPv4 addressing
LOCLADDR6	The LOCLADDR parameter is set to IPv6 addressing
IPADDR4	IPADDRV is set to IPv4 addressing
IPADDR6	IPADDRV is set to IPv6 addressing

Table 98. System configurations

Originating queue manager		Responding queue manager			Result
Queue manager and Stack	**LOCLADDR**	**IPADDRV**	**Queue Manager and Stack**	**DNS Return**	
v71 + IPv6	Any	Not applicable			IP Error
v71 + IPv4 or v71 + Dual	Both LOCLADDR4 & LOCLADDR6	Not applicable	v71 + IPv4 or v71 + Dual	IPv4DNS or DualDNS	IPv4 connection can be established
v71 + IPv4 or v71 + Dual	Blank or LOCLADDR4	Not applicable	v71 + IPv4 or v71 + Dual	IPv4DNS or DualDNS	IPv4 connection can be established
v71 + IPv4 or v71 + Dual	Blank or LOCLADDR4	Not applicable	v71 + Dual	IPv6DNS	Unable to resolve CONNAME
v71 + IPv4 or v71 + Dual	Blank or LOCLADDR4	Not applicable	v71 + Dual or v8 + Dual v8 + IPv4	IPv4DNS or DualDNS	IPv4 connection can be established
v71 + IPv4 or v71 + Dual	LOCLADDR6	Not applicable			IP Error
v71 + IPv4 or v71 + Dual	Blank or LOCLADDR4 or both LOCLADDR4 & LOCLADDR6	Not applicable	v8 + IPv6	IPv6DNS	Unable to resolve CONNAME

Table 98. System configurations (continued)

Originating queue manager		**Responding queue manager**			**Result**
Queue manager and Stack	**LOCLADDR**	**IPADDRV**	**Queue Manager and Stack**	**DNS Return**	
v8 + IPv4	Blank or LOCLADDR4	Not specified	v71 + IPv4 or v71 + Dual or v8 + IPv4	IPv4DNS or DualDNS	IPv4 connection can be established
v8 + IPv4	LOCADD6	Not specified			Unable to resolve LOCLADDR
v8 + IPv4	Blank or LOCLADDR4	Not specified	v8 + IPv6	IPv6DNS	Unable to resolve CONNAME
v8 + IPv6	Blank or LOCLADDR6	Not specified	v71 + Dual	DualDNS	Attempts to start IPv6 channel and fails as there will be no IPv6 listener available
v8 + IPv6	Blank or LOCLADDR6	Not specified	v71 + IPv4	IPv4DNS	Attempts to start IPv6 channel and fails as there will be no IPv6 listener available
v8 + IPv6 or v8 + Dual	LOCLADDR6	Blank or IPADDR6	v8 + IPv6 or v8 + Dual	IPv6DNS or DualDNS	IPv6 connection can be established
v8 + Dual	LOCLADDR6	IPADDR4	v8 + Dual	IPv4DNS or DualDNS	IPv6 connection can be established where mapped addressing can be used
v8 + Dual	Blank or LOCLADDR4	IPADDR4	v71 + Dual	IPv4DNS or DualDNS	IPv4 connection can be established
v8 + Dual	Both LOCLADDR4 & LOCLADDR6	Blank or IPADDR4	v71 + Dual	IPv4DNS or DualDNS	IPv4 connection can be established
v8 + Dual	LOCLADDR4	IPADDR4			Unable to resolve LOCLADDR
v8 + Dual	LOCLADDR6 or both LOCLADDR4 & LOCLADDR6	Blank or IPADDR6	v8 + IPv6 or v8 + Dual	IPv6DNS or DualDNS	IPv6 connection can be established

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.

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

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

For instructions on how to create, start, stop, and delete a queue manager, see "Creating and managing queue managers on Multiplatforms."

For information about how to create the components required to connect your IBM MQ installations and applications together, see "Configuring distributed queuing" on page 961.

For instructions on how to connect your clients to an IBM MQ server by using different methods, see "Configuring connections between the server and client" on page 844.

For instructions on how to configure a queue manager cluster, see "Configuring a queue manager cluster" on page 1063.

You can change the behavior of IBM MQ or a queue manager by changing configuration information. For more information, see "Changing IBM MQ and queue manager configuration information" on page 906. In general, you do not need to restart a queue manager for any configuration changes to take effect, except for when stated in this product documentation.

z/OS For instructions on how to configure IBM MQ for z/OS, see "Configuring queue managers on z/OS" on page 1455.

Related tasks:

z/OS "Configuring queue managers on z/OS" on page 1455
Use these instructions to configure queue managers on IBM MQ for z/OS.

Related information:

IBM MQ technical overview

Administering local IBM MQ objects

Administering remote IBM MQ objects

IBM i Administering IBMi

z/OS Administering IBM MQ for z/OS

Planning

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

Creating and managing queue managers on Multiplatforms

Multi

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 UNIX, 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.

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

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

Specify a default queue manager
Each node should have a default queue manager, though it is possible to configure IBM MQ on a

node without one. The default queue manager is the queue manager that applications connect to if they do not specify a queue manager name in an `MQCONN` call. It is also the queue manager that processes MQSC commands when you invoke the `runmqsc` command without specifying a queue manager name.

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

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

Creating a default queue manager is described in "Creating a default queue manager" on page 836.

Specify a dead-letter queue

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

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

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

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

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

Specify a default transmission queue

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

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

Specify the logging parameters you require

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

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

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

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

For more information about logging parameters, see “Configuring high availability, recovery and restart” on page 1198.

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

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”
- “Making an existing queue manager the default” on page 838
- “Backing up configuration files after creating a queue manager” on page 839
- “Starting a queue manager” on page 840
- “Stopping a queue manager” on page 841
- “Restarting a queue manager” on page 842
- “Deleting a queue manager” on page 843

Related tasks:

“Changing IBM MQ and queue manager configuration information” on page 906
You can change the behavior of IBM MQ or an individual queue manager to suit the needs of your installation.

z/OS “Configuring queue managers on z/OS” on page 1455
Use these instructions to configure queue managers on IBM MQ for z/OS.

Related information:

Creating a queue manager called QM1

System and default objects

crtmqm

Creating a default queue manager

Multi

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

Before you begin

Before creating a default queue manager, read through the considerations described in "Creating and managing queue managers on Multiplatforms" on page 833.

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

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

About this task

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

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

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

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

Procedure

To create a default queue manager, use the **crtmqm** command with the **-q** flag. The following example of the **crtmqm** command creates a default queue manager called SATURN.QUEUE.MANAGER:

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

where:

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

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

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

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

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

What to do next

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

Related tasks:

"Backing up configuration files after creating a queue manager" on page 839
IBM MQ configuration information is stored in configuration files on UNIX, 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.

Related information:

Working with queue managers

Working with local queues

System and default objects

Making an existing queue manager the default

▶ Multi

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.

▶ Linux ▶ Windows 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 839.

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

Procedure

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

Related tasks:
"Changing IBM MQ and queue manager configuration information" on page 906
You can change the behavior of IBM MQ or an individual queue manager to suit the needs of your installation.

Using IBM MQ Explorer to make a queue manager the default

Linux Windows

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 838.

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

ULW

IBM MQ configuration information is stored in configuration files on UNIX, 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.

V 9.0.0 If you have installed the AMQP service, then there is an additional configuration file that you must back up:

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

Related tasks:
"Changing IBM MQ and queue manager configuration information" on page 906
You can change the behavior of IBM MQ or an individual queue manager to suit the needs of your installation.
"Backing up and restoring IBM MQ queue manager data" on page 1352
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.

Starting a queue manager

Multi

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

About this task

You can start a queue manager by using the **strmqm** command. For a description of the **strmqm** command and its options, see strmqm.

Windows Linux Alternatively, on Windows and Linux (x86 and x86-64 platforms) systems, you can start a queue manager by using the IBM MQ Explorer.

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

Procedure

- To start a queue manager by using the **strmqm** command, enter the command followed by the name of the queue manager that you want to start. For example, to start a queue manager called `QMB`, enter the following command:
  ```
  strmqm QMB
  ```
  Note: You must use the **strmqm** command from the installation associated with the queue manager that you are working with. You can find out which installation a queue manager is associated with using the `dspmq -o installation` command.

  The `strmqm` command does not return control until the queue manager has started and is ready to accept connection requests.
- Windows Linux To start a queue manager by using the IBM MQ Explorer, complete the following steps:
  1. Open the IBM MQ Explorer.
  2. In the Navigator view, select the queue manager.
  3. Click **Start**.

Results

The queue manager starts.

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

Stopping a queue manager

Multi

You can use the **endmqm** command to stop a queue manager. This command provides three ways to stop a queue manager: a controlled, or quiesced, shutdown, an immediate shutdown, and a preemptive shutdown. Alternatively, on Windows and Linux, you can stop a queue manager by using the IBM MQ Explorer.

About this task

There are three ways to stop a queue manager with the **endmqm** command:

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

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

Preemptive shutdown
The queue manager stops immediately. Use this type of shutdown only in exceptional circumstances, for example, when a queue manager does not stop as a result of a normal **endmqm** command.

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

Tip: Problems with shutting down a queue manager are often caused by applications. For example, when applications:
- Do not check MQI return codes properly
- Do not request notification of a quiesce
- Terminate without disconnecting from the queue manager (by issuing an `MQDISC` call)

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

Windows Linux As an alternative to using the **endmqm** command, on Windows and Linux, you can stop a queue manager by using the IBM MQ Explorer to carry out either a controlled or an immediate shutdown.

Procedure

- To stop the queue manager by using the **endmqm** command, enter the command followed by the parameter, if required, and the name of the queue manager that you want to stop.

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

  – To carry out a controlled (quiesced) shutdown, enter the **endmqm** command as shown in the following example, which stops a queue manager called `QMB`:

  ```
  endmqm QMB
  ```

  Alternatively, entering the **endmqm** command with the **-c** parameter , as shown in the following example, is equivalent to an `endmqm QMB` command.

  ```
  endmqm -c QMB
  ```

In both cases, control is returned to you immediately and you are not notified when the queue manager has stopped. If you want the command to wait until all applications have stopped and the queue manager has ended before returning control to you, use the **-w** parameter instead as shown in the following example.

```
endmqm -w QMB
```

– To carry out an immediate shutdown, enter the **endmqm** command with the **-i** parameter as shown in the following example:

```
endmqm -i QMB
```

– To carry out a preemptive shutdown, enter the **endmqm** command with the **-p** parameter as shown in the following example:

```
endmqm -p QMB
```

Attention: A preemptive shutdown can have unpredictable consequences for connected applications. Do not use this option unless all other attempts to stop the queue manager by using a normal **endmqm** command have failed. ULW If the preemptive shutdown does not work, try stopping the queue manager manually instead.

- Windows Linux On Windows and Linux, to stop the queue manager by using IBM MQ Explorer, complete the following steps:
  1. Open the IBM MQ Explorer.
  2. Select the queue manager from the Navigator View.
  3. Click **Stop**. The End Queue Manager panel is displayed.
  4. Select **Controlled**, or **Immediate**.
  5. Click **OK**. The queue manager stops.

Related information:

Applying maintenance level updates to multi-instance queue managers on Windows

Applying maintenance level updates to multi-instance queue managers on UNIX and Linux

Restarting a queue manager

Multi

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

About this task

You can restart a queue manager by using the **strmqm** command. For a description of the **strmqm** command and its options, see strmqm.

Windows Linux On Windows and Linux x86-64 systems, you can restart a queue manager by using the IBM MQ Explorer in the same way as for starting a queue manager.

Procedure

- To restart a queue manager by using the **strmqm** command, enter the command followed by the name of the queue manager that you want to restart. For example, to start a queue manager called `strmqm saturn.queue.manager`, enter the following command:

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

- Windows Linux To start a queue manager by using the IBM MQ Explorer, complete the following steps:
  1. Open the IBM MQ Explorer.

2. In the Navigator view, select the queue manager.
3. Click **Start**.

Results

The queue manager restarts.

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

Deleting a queue manager

Multi

You can delete a queue manager using the `dltmqm` command. Alternatively, on Windows and Linux systems, you can use the IBM MQ Explorer to delete a queue manager.

Before you begin

Attention:

- Deleting a queue manager is a drastic step, because you also delete all resources associated with the queue manager, including all queues and their messages and all object definitions. If you use the `dltmqm` command, there is no displayed prompt that allows you to change your mind; when you press the Enter key all the associated resources are lost.
- Windows On Windows, deleting a queue manager also removes the queue manager from the automatic startup list (described in "Starting a queue manager" on page 840). When the command has completed, an `IBM MQ queue manager ending` message is displayed; you are not told that the queue manager has been deleted.
- Deleting a cluster queue manager does not remove it from the cluster. For more information, see the usage notes in dltmqm.

About this task

You can delete a queue manager by using the `dltmqm` command. For a description of the `dltmqm` command and its options, see dltmqm. Ensure that only trusted administrators have the authority to use this command. (For information about security, see Setting up security on UNIX, Linux, and Windows.)

Windows Linux Alternatively, on Windows and Linux (x86 and x86-64 platforms) systems, you can delete a queue manager by using the IBM MQ Explorer.

Procedure

- To delete a queue manager by using the `dltmqm` command, complete the following steps:
  1. Stop the queue manager.
  2. Issue the following command:
     ```
     dltmqm QMB
     ```
     Note: You must use the `dltmqm` command from the installation associated with the queue manager that you are working with. You can find out which installation a queue manager is associated with using the `dspmq -o installation` command.
- Windows Linux To delete a queue manager by using the IBM MQ Explorer, complete the following steps:
  1. Open the IBM MQ Explorer.

2. In the Navigator view, select the queue manager.
3. If the queue manager is not stopped, stop it. To stop the queue manager, right-click it and then click **Stop**.
4. Delete the queue manager. To delete the queue manager, right-click it and then click **Delete**.

Results

The queue manager is deleted.

Configuring connections between the server and client

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.

Procedure

1. Decide on the form of communication that you are going to use. See “Which communication type to use” on page 845.
2. Define the connection at each end of the channel. To define the connection, you must:
   a. Configure the connection.
   b. Record the values of the parameters that you need for the channel definitions.
   c. Enable the server to detect incoming network requests from your IBM MQ MQI client, by starting a *listener*.

Related concepts:

“Configuring an extended transactional client” on page 847
This collection of topics describes how to configure the extended transactional function for each category of transaction manager.

“Defining MQI channels” on page 858
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*.

“Channel-exit programs for MQI channels” on page 870
Three types of channel exit are available to the IBM MQ MQI client environment on UNIX, Linux, and Windows.

Related tasks:

“Creating server-connection and client-connection definitions on different platforms” on page 859
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.

“Creating server-connection and client-connection definitions on the server” on page 862
You can create both definitions on the server, then make the client-connection definition available to the client.

z/OS "Connecting a client to a queue-sharing group" on page 875
You can connect a client to a queue-sharing group by creating an MQI channel between a client and a queue manager on a server that is a member of a queue-sharing group.

"Configuring a client using a configuration file" on page 876
You 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 information:

Connecting IBM MQ MQI client applications to queue managers

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

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

Note:

1. Except Linux (POWER platform)

Related concepts:

"Defining a TCP connection on Windows" on page 1026
Define a TCP connection by configuring a channel at the sending end to specify the address of the target, and by running a listener program at the receiving end.

"Defining a TCP connection on UNIX and Linux" on page 1034
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.

IBM i "Defining a TCP connection on IBM i" on page 1055
You can define a TCP connection within the channel definition using the Connection Name field.

z/OS "Defining a TCP connection on z/OS" on page 1554
To define a TCP connection, there are a number of settings to configure.

"Defining an LU 6.2 connection on Windows" on page 1028
SNA must be configured so that an LU 6.2 conversation can be established between the two machines.

"Defining an LU 6.2 connection on UNIX and Linux" on page 1038
SNA must be configured so that an LU 6.2 conversation can be established between the two machines.

IBM i "Defining an LU 6.2 connection on IBM i" on page 1057
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 1030
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 reference:

"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.

Related information:

z/OS "Defining an LU6.2 connection for z/OS using APPC/MVS" on page 1557
To define an LU6.2 connection there are a number of settings to configure.

Defining a TCP/IP connection

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

This information has moved. See the following topics:

- "Which communication type to use" on page 845
- Windows "Defining a TCP connection on Windows" on page 1026
- UNIX Linux "Defining a TCP connection on UNIX and Linux" on page 1034
- IBM i "Defining a TCP connection on IBM i" on page 1055
- z/OS "Defining a TCP connection on z/OS" on page 1554

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

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

If the 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/AMQERROn.LOG` on UNIX and Linux systems or `amqerr0n.log` in the errors subdirectory of the IBM MQ client installation on Windows). If the client retries the `MQCONN` request, it might be successful.

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

Defining a NetBIOS connection

NetBIOS connections apply only to Windows systems.

This information has moved. See “Which communication type to use” on page 845 and “Defining a NetBIOS connection on Windows” on page 1030.

Configuring an extended transactional client

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

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

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

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

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

Related concepts:

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

"Microsoft Transaction Server" on page 857
No additional configuration is required before you can use MTS as a transaction manager. However, there are some points to note.

Configuring XA-compliant transaction managers

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

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

To configure an extended transactional client, you must first configure the IBM MQ base client as described in:

- AIX Installing an IBM MQ client on AIX
- HP-UX Installing an IBM MQ client on HP-UX
- Linux Installing an IBM MQ client on Linux
- Solaris Installing an IBM MQ client on Solaris
- Windows Installing an IBM MQ client on Windows
- IBM i 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 850.

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 854.

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 856
- "Configuring an extended transactional client for Tuxedo" on page 857

Related concepts:

"The CHANNEL, TRPTYPE, CONNAME, and QMNAME parameters of the xa_open string" on page 852
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 854
The xa_open call fails in certain circumstances.

Related tasks:

"Using the extended transactional client with TLS channels" on page 855
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 853
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 854
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: z/OS

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.

For transaction managers which encounter this situation, take the following actions to allow the transaction manager to interact with IBM MQ for z/OS:
- Apply the fix for APAR PI73140.
- Enable the change provided by PI73140 for the server-connection channel used by the transaction manager.

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

Note that channels with the CSQSERVICE1 keyword have the following restrictions:
- GROUP unit of recovery disposition is not permitted. Only QMGR unit of recovery disposition is allowed.

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

Related concepts:
"Configuring XA-compliant transaction managers" on page 848
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.

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

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

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

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

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

This is an optional parameter.

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

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

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

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

Here is an example of an xa_open string:

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

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

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

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

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

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

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

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

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

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

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

The TPM and AXLIB parameters:

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

If the TPM parameter is supplied in an xa_open string, but the AXLIB parameter is not supplied, the extended transactional client assumes a value for the AXLIB parameter based on the value of the TPM parameter. See Table 101 for the assumed values of the AXLIB parameter.

Table 101. Assumed values of the AXLIB parameter

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

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

Additional error processing for xa_open:

The xa_open call fails in certain circumstances.

Topics in this section describe situations in which the xa_open call fails. It also fails if any of the following situations occur:
- There are errors in the xa_open string.
- There is insufficient information to start an MQI channel.
- There is a problem while trying to start an MQI channel (the server queue manager is not running, for example).

Recovery following a failure in extended transactional processing:

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

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

Alternatively, you must ensure that the configuration changes do not affect the ability of the transaction manager to open the queue managers it needs to open. 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 102.

Table 102. IBM MQ libraries containing the XA switch structures

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

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

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

Related concepts:

"Dynamic registration and extended transactional processing"
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, HP-UX, Solaris, and Windows systems. Each switch load file contains a function that returns a pointer to the XA switch structure that is used for dynamic registration, MQRMIXASwitchDynamic. See Table 103 for the fully qualified path name of each switch load file.

Table 103. The switch load files

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

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

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

```
cicsadd -c xad -r REGION1 WMQXA \
    ResourceDescription="IBM MQ queue manager MARS" \
    XAOpen="channel=MARS.SVR,trptype=tcp,conname=MARS(1415),qmname=MARS,tpm=cics" \
    SwitchLoadFile="C:\Program Files\IBM\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 104 on page 857. 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 104. CICS task termination exits

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

Configuring an extended transactional client for Tuxedo:

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

To configure XAD resource definition for use by Tuxedo, perform the following actions:
- In the GROUPS section of the Tuxedo UBBCONFIG file for an application, use the OPENINFO parameter to specify an xa_open string.

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

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

Microsoft Transaction Server

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

Note the following information about using MTS with the extended transactional client:
- An MTS application always starts an MQI channel when it connects to a server queue manager. MTS, in its role as a transaction manager, then uses the same MQI channel to communicate with the queue manager.
- Following a failure, MTS must be able to recover any incomplete units of work. To do this, MTS must be able to communicate with any queue manager that was participating in an incomplete unit of work at the time of the failure.

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

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

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

  If you ever need to change any configuration information, therefore, ensure that all incomplete units of work have been resolved before making the changes. Alternatively, ensure that the configuration

changes do not affect the ability of the extended transactional client to restart an MQI channel using the information recorded by MTS. 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 859 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 862 for details on how to use this method.

Automatically defined channels

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

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

Related concepts:

"Channel control function" on page 991
The channel control function provides facilities for you to define, monitor, and control channels.

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.

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

IBM i On IBM i you can also use the panel interface.

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

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 860.
- To create a client-connection channel on an IBM MQ MQI client, see "Creating a client-connection channel on the IBM MQ MQI client" on page 860.

Defining a server-connection channel on the server

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

Procedure

1. Optional: If your server platform is not z/OS, first create and start a queue manager and then start MQSC commands.
   a. Create a queue manager, called QM1 for example:
      ```
      crtmqm QM1
      ```
   b. Start the queue manager:
      ```
      strmqm QM1
      ```
   c. Start MQSC commands:
      ```
      runmqsc QM1
      ```
2. Define a channel with your chosen name and a channel type of *server-connection*.
   ```
   DEFINE CHANNEL(CHAN1) CHLTYPE(SVRCONN) TRPTYPE(TCP) +
   DESCR('Server-connection to Client_1')
   ```
   This channel definition is associated with the queue manager running on the server.
3. Use the following command to allow the inbound connect access to your queue manager:
   ```
   SET CHLAUTH(CHAN1) TYPE(ADDRESSMAP) ADDRESS('IP address') MCAUSER('userid')
   ```
   - Where SET CHLAUTH uses the name of the channel defined in the previous step.
   - Where *'IP address'* is the IP address of the client.
   - Where *'userid'* is the ID you want to provide to the channel for access control to the target queues. This field is case-sensitive.

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

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

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

Using MQSERVER

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

- Specify a simple channel definition on Windows as follows:
  ```
  SET MQSERVER=ChannelName/TransportType/ConnectionName
  ```
- Specify a simple channel definition on UNIX and Linux systems as follows:
  ```
  export MQSERVER=ChannelName/TransportType/ConnectionName
  ```
- Specify a simple channel definition on IBM i systems as follows:
  ```
  ADDENVVAR ENVVAR(MQSERVER) VALUE('ChannelName/TransportType/ConnectionName')
  ```

where:

- `ChannelName` must be the same name as defined on the server. It cannot contain a forward slash.
- `TransportType` can be one of the following values, depending on your IBM MQ MQI client platform:
  - LU62
  - TCP
  - NETBIOS
  - SPX

Note: On UNIX and Linux systems, the TransportType is case-sensitive and must be uppercase. An `MQCONN` or `MQCONNX` call returns 2058 if the TransportType is not recognized

- `ConnectionName` is the name of the server as defined to the communications protocol (TransportType).

For example, on Windows:

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

or, on UNIX and Linux systems:

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

Note: To change the TCP/IP port number, see "MQSERVER" on page 902.

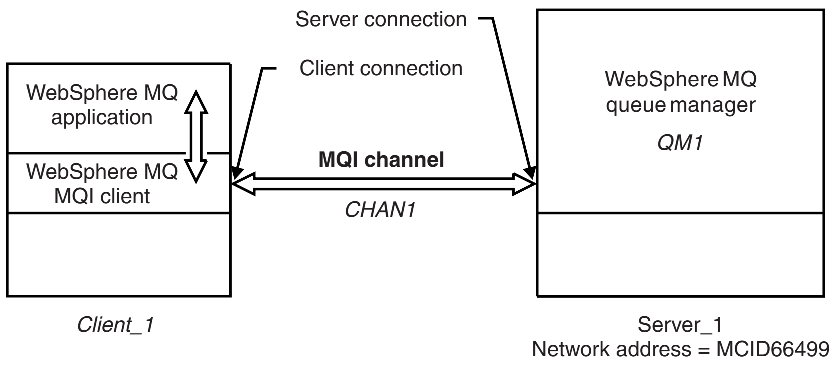


Figure 76. Simple channel definition

Some more examples of simple channel definitions are as follows:

- On Windows:

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

- On UNIX and Linux systems:

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

- IBM i On IBM i:

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

where `BOX99` is the LU 6.2 ConnectionName.

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

Note: For more information about the *MQSERVER* environment variable, see "MQSERVER" on page 902.

Using the MQCNO structure on an MQCONNX call

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

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

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

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

If a channel uses 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 **MQCONNX** call.

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

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

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

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

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.
- Linux Windows On Linux and Windows, you can also use IBM MQ Explorer.
- z/OS On z/OS, you can also use the Operation and Control panels.
- IBM i On IBM i you can also use the panel interface.

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

Procedure

1. To define a server-connection channel, see "Defining the server-connection channel on the server" on page 867.
2. To define a client-connection channel, see "Defining the client-connection channel on the server" on page 868.

Related concepts:

"Client channel definition table"
The client channel definition table (CCDT) determines the channel definitions and authentication information used by client applications to connect to the queue manager. On Multiplatforms, a CCDT is created automatically. You must then make it available to the client application.

Related tasks:

"Defining the server-connection channel on the server" on page 867
Create a server-connection channel definition for the queue manager.

"Defining the client-connection channel on the server" on page 868
Having defined the server-connection channel, you now define the corresponding client-connection channel.

"Accessing client-connection channel definitions" on page 869
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. V 9.0.0 From Version 9.0, IBM MQ also provides the ability to locate a client channel definition table (CCDT) through a URL.

Client channel definition table

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

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

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

Multi On Multiplatforms, the CCDT is created when the queue manager is created. The CCDT associated with a queue manager is kept in sync with the object definitions, therefore 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.

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

Multi

Default CCDT `AMQCLCHL.TAB`

On Multiplatforms, a default CCDT called `AMQCLCHL.TAB` is created when you create a queue manager.

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

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

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

- UNIX Linux On UNIX and Linux systems:

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

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

- Windows On Windows:

  MQ_INSTALLATION_PATH\data\qmgrs\QUEUEMANAGERNAME\@ipcc

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

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

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

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

Locations for the client channel definition table

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

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

V 9.0.0 From IBM MQ Version 9.0, the CCDT can be hosted in a central location that is accessible through a URI, removing the need to individually update the CCDT for each deployed client. Version 9.0 adds the capability for native (C/C++, COBOL and RPG) and unmanaged .NET applications to pull the CCDT from a URL, whether that be a local file, ftp or http resource. 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/CCDTUrlOffset via MQCNO structure being passed into MQCONNX MQI call
- MQCCDTURL environment variable
- ChannelDefinitionDirectory attribute in the Channels stanza of mqclient.ini

V 9.0.0 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
```

V 9.0.0 If you wanted to use this support with ftp or http, then that still means that you would need to host the CCDT file on a server, but with the support added at Version 9.0, all of your client applications could 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 “Web addressable access to the client channel definition table” on page 865.

How to use runmqsc to create a CCDT directly on a client machine

From IBM MQ Version 8.0, you can create a CCDT on the client machine directly by using 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.

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

How to specify the location of the CCDT on the client

On a client system, you can specify the location of the CCDT in 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 attributes `ChannelDefinitionDirectory` to specify the directory where the table is located, and `ChannelDefinitionFile` to specify the file name.
- V 9.0.0 By providing a URL (file, ftp, or http) for a CCDT that is hosted in a central location (see “Locations for the client channel definition table” on page 864.

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.

V 9.0.0 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 “Web addressable access to the client channel definition table.”

Related reference:

“MQCHLLIB” on page 899
`MQCHLLIB` specifies the directory path to the file containing the client channel definition table (CCDT). The file is created on the server, but can be copied across to the IBM MQ MQI client workstation.

Related information:

Working with revoked certificates

Web addressable access to the client channel definition table

V 9.0.0

From Version 9.0, IBM MQ provides the ability to locate a client channel definition table (CCDT) through a URL, either by programming using MQCNO, using environment variables, or using `mqclient.ini` file stanzas.

Attention: You can use the environment variable option 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.

The environment variable “MQCCDTURL” on page 897 allows you to provide a file, ftp, or http URL as a single value from which a client channel definition table can be obtained.

You can also use “MQCHLLIB” on page 899 (or that specified by **`ChannelDefinitionDirectory`** under the “CHANNELS stanza of the client configuration file” on page 883) to locate a CCDT file, either through file, ftp, or http URL, in addition to the existing local file system directory, that is, `/var/mqm`).

Note that an “MQCHLLIB” on page 899 value is a directory stem and works in combination with “MQCHLTAB” on page 901 to derive the fully qualified URL.

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" on page 902 environment variable.
4. If an `mqclient.ini` file is defined and contains a ServerConnectionParms then the channel that it defines is used. For more information, see "Configuring a client using a configuration file" on page 876 and "CHANNELS stanza of the client configuration file" on page 883.
5. "MQCCDTURL" on page 897 environment variable.
6. "MQCHLLIB" on page 899 and "MQCHLTAB" on page 901 environment variable.
7. **ChannelDefinitionDirectory** in the "CHANNELS stanza of the client configuration file" on page 883.

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 **runmqsc** DEFINE CHANNEL 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 concepts:

"Client channel definition table" on page 863
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 CCDT is created automatically. You must then make it available to the client application.

Related tasks:

"Accessing client-connection channel definitions" on page 869
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. V 9.0.0 From Version 9.0, IBM MQ also provides the ability to locate a client channel definition table (CCDT) through a URL.

Related information:

CCDTURL

Using a CCDT with IBM MQ classes for JMS

XMSC_WMQ_CCDTURL

Client connection channels in the Active Directory

Windows

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

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

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

Defining the server-connection channel on the server

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

Procedure

1. On the server machine, define a channel with your chosen name and a channel type of *server-connection*. For example:
   ```
   DEFINE CHANNEL(CHAN2) CHLTYPE(SVRCONN) TRPTYPE(TCP) +
   DESCR('Server-connection to Client_2')
   ```
2. Use the following command to allow the inbound connect access to your queue manager:
   ```
   SET CHLAUTH(CHAN2) TYPE(ADDRESSMAP) ADDRESS('IP address') MCAUSER('userid')
   ```
   - Where SET CHLAUTH uses the name of the channel defined in the previous step.
   - Where *'IP address'* IP address is the IP address of the client.
   - Where *'userid'* is the ID you want to provide to the channel for access control to the target queues. This field is case-sensitive.

   You can choose to identify your inbound connection using a number of different attributes. The example uses IP address. Alternative attributes include client user ID and TLS Subject Distinguished Name. For more information, see Channel authentication records This channel definition is associated with the queue manager running on the server.

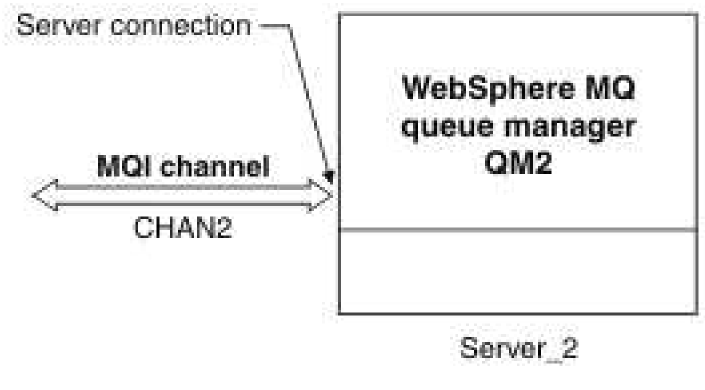


Figure 77. Defining the server-connection channel

Defining the client-connection channel on the server

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

Before you begin

Define the server-connection channel.

Procedure

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

   ```
   DEFINE CHANNEL(CHAN2) CHLTYPE(CLNTCONN) TRPTYPE(TCP) +
   CONNAME(9.20.4.26) QMNAME(QM2) DESCR('Client-connection to Server_2')
   ```
2. Use the following command to allow the inbound connect access to your queue manager:

   ```
   SET CHLAUTH(CHAN2) TYPE(ADDRESSMAP) ADDRESS('IP-address') MCAUSER('userid')
   ```
   - Where SET CHLAUTH uses the name of the channel defined in the previous step.
   - Where *'IP address'* is the IP address of the client.
   - Where *'userid'* is the ID you want to provide to the channel for access control to the target queues. This field is case-sensitive.

   You can choose to identify your inbound connection using a number of different attributes. The example uses IP address. Alternative attributes include client user ID and 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 "Client channel definition table" on page 863.

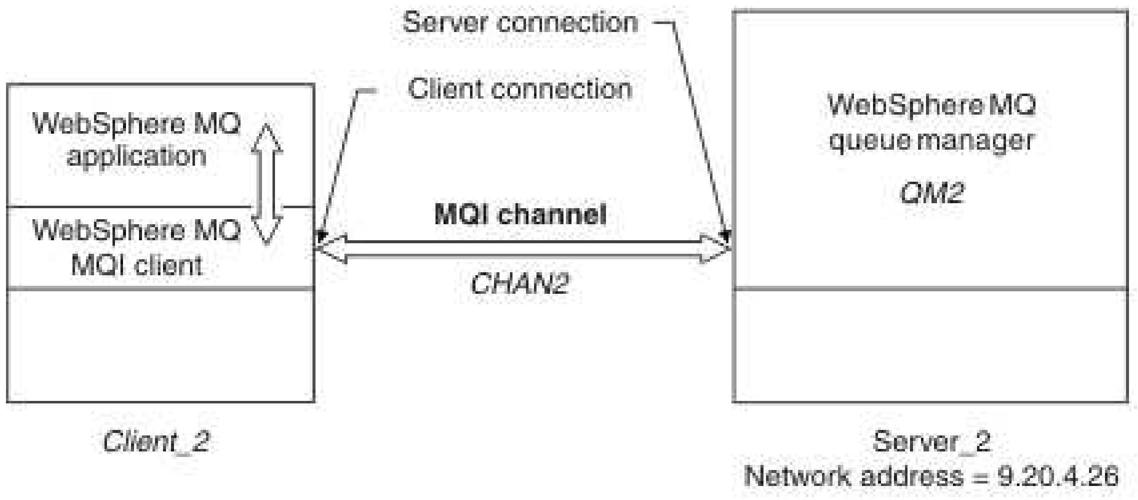


Figure 78. Defining the client-connection channel

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. V 9.0.0 From Version 9.0, IBM MQ also provides the ability to locate a client channel definition table (CCDT) through a URL.

Before you begin

You have defined the client-connection channels that you need.

z/OS On z/OS, you have created a CCDT.

Multi On Multiplatforms, the CCDT is automatically created and updated.

About this task

For a client application to use the client channel definition table (CCDT), you must make the CCDT available to it and specify its location and name. There are several ways of doing this:

- You can copy the CCDT to the client computer.
- You can copy the CCDT to a location shared by more than one client.
- You can make the CCDT accessible to the client as a shared file, while it remains located on the server.

V 9.0.0 From Version 9.0, IBM MQ, native (C/C++, COBOL and RPG) and unmanaged .NET applications can pull the CCDT hosted in a central location from a URL, whether that be a local file, ftp or http resource.

Procedure

1. Make the CCDT available to the client applications 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. V 9.0.0 Optional: Define a local file, ftp or http URL for a CCDT hosted in a central location so that native (C/C++, COBOL and RPG) and unmanaged .NET applications can pull the CCDT from this URL.

Whichever location you choose for the CCDT, the location must be secure to prevent unauthorized changes to the channels.

2. On the client, specify the location and name of the file containing the CCDT in one of three ways:
   a. Optional: Use the CHANNELS stanza of the client configuration file. For more information, see "CHANNELS stanza of the client configuration file" on page 883.
   b. Optional: Use the environment variables MQCHLLIB and MQCHLTAB.

      For example, you can set the environment variables by typing:
      - On NSS Client HP Integrity NonStop Server, and UNIX and Linux systems:
        ```
        export MQCHLLIB= MQ_INSTALLATION_PATH/qmgrs/ QUEUEMANAGERNAME /@ipcc
        export MQCHLTAB=AMQCLCHL.TAB
        ```
      - IBM i On IBM i:
        ```
        ADDENVVAR ENVVAR(MQCHLLIB) VALUE('/QIBM/UserData/mqm/qmgrs/QUEUEMANAGERNAME/@ipcc')
        ADDENVVAR ENVVAR(MQCHLTAB) VALUE(AMQCLCHL.TAB)
        ```

      where *MQ_INSTALLATION_PATH* represents the high-level directory in which IBM MQ is installed.
   c. Optional: On Windows only, use the **setmqscp** control command to publish the client-connection channel definitions in Active Directory.
   d. V 9.0.0 Provide the location of a centrally hosted CCDT through a URL, either by programming using MQCNO, using environment variables, or using `mqclient.ini` file stanzas. For more information, see "Locations for the client channel definition table" on page 864 and "Web addressable access to the client channel definition table" on page 865.

   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 concepts:

"Client channel definition table" on page 863
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 CCDT is created automatically. You must then make it available to the client application.

V 9.0.0 "Web addressable access to the client channel definition table" on page 865
From Version 9.0, IBM MQ provides the ability to locate a client channel definition table (CCDT) through a URL, either by programming using MQCNO, using environment variables, or using `mqclient.ini` file stanzas.

Related information:

MQI client: Client Channel Definition Table (CCDT)

Channel-exit programs for MQI channels

ULW

Three types of channel exit are available to the IBM MQ MQI client environment on UNIX, 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 872
You can use security exit programs to verify that the partner at the other end of a channel is genuine. Special considerations apply when a security exit is applied to a client connection.

Related reference:

"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.

"Identifying the API call in a send or receive exit program" on page 873
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.

Related information:

Extending queue manager facilities

User exits, API exits, and IBM MQ installable services

Path to exits

ULW

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

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

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

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

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

Security exits on a client connection

ULW

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 79 on page 873 illustrates the use of security exits in a client connection, using the IBM MQ object authority manager to authenticate a user. Either SecurityParmsPtr or SecurityParmsOffset is set in the MQCNO structure on the client and there are security exits at both ends of the channel. After the normal security message exchange has ended, and the channel is ready to run, the MQCSP structure accessed from the MQCXP SecurityParms field is passed to the security exit on the client. The exit type is set to MQXR_SEC_PARMS. The security exit can elect to do nothing to the user identifier and password, or it can alter either or both of them. The data returned from the exit is then sent to the server-connection end of the channel. The MQCSP structure is rebuilt on the server-connection end of the channel and is passed to the server-connection security exit accessed from the MQCXP SecurityParms field. The security exit receives and processes this data. This processing is typically to reverse any change made to the user ID and password fields in the client exit, which are then used to authorize the queue manager connection. The resulting MQCSP structure is referenced using SecurityParmsPtr in the MQCNO structure on the queue manager system.

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

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

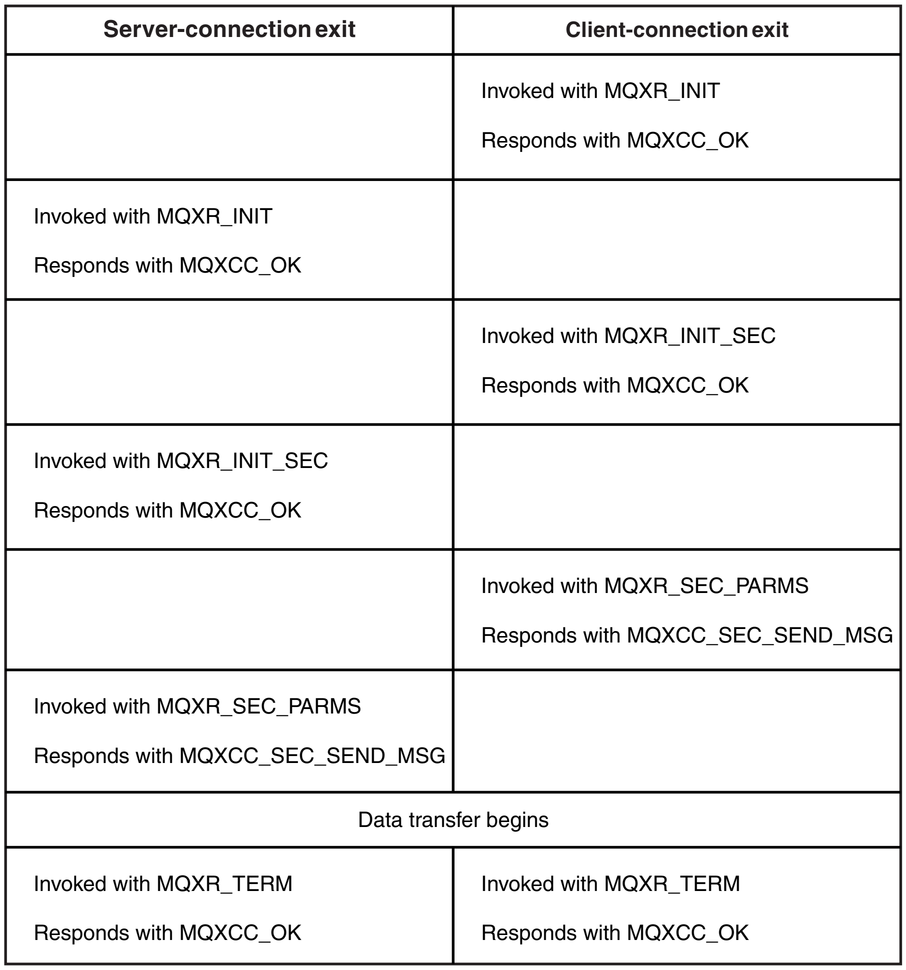
If SecurityParmsPtr and SecurityParmsOffset are not set in the MQCNO structure and there is a security exit at either or both ends of the channel, the security exit or exits are called. Either security exit can return its own MQCSP structure, addressed through the SecurityParmsPtr; the security exit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

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

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

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

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



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

Figure 79. 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 Version 7.1 might require updating. For more information see Channel security exit programs.

Identifying the API call in a send or receive exit program

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

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

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

Table 105. Identifying API calls

API call	Value of byte 10 for request	Value of byte 10 for reply
MQCONN [1, 2]	X'81'	X'91'
MQDISC [1]	X'82'	X'92'
MQOPEN [3]	X'83'	X'93'
MQCLOSE	X'84'	X'94'
MQGET [4]	X'85'	X'95'
MQPUT [4]	X'86'	X'96'
MQPUT1 request [4]	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

z/OS

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 MQCONNX 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.

Configuring a client using a configuration file

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

About this task

You configure your IBM MQ MQI clients by using a text file, similar to the queue manager configuration file, `qm.ini`, used on UNIX and Linux platforms. 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 all platforms, and 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 878.

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=32768
   ClntRcvBuffSize=32768
   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)
```

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

Environment variables that were supported in releases of IBM WebSphere MQ earlier than Version 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.

Procedure

Use the information in the following topics to help with configuring your clients:
- "Location of the client configuration file"
- "Which IBM MQ clients can read each attribute" on page 878

Location of the client configuration file

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

A client application uses the following search path to locate the IBM MQ MQI client configuration file:
1. The location specified by the environment variable MQCLNTCF.

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

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

   Note the following:
   - The IBM MQ data directory does not exist on certain platforms, for example, IBM i and z/OS, or in cases where the client has been supplied with another product.
   - On UNIX and Linux systems, the directory is `/var/mqm`
   - On Windows platforms you configure the environment variable MQ_FILE_PATH during installation, to point at the data directory. It is normally `C:\ProgramData\IBM\MQ`
   - To allow for Java 1.4.2 JREs that do not support reading environment variables you can manually override the MQ_FILE_PATH environment variable with an MQ_FILE_PATH Java System Property.
4. A file called `mqclient.ini` in a standard directory appropriate to the platform, and accessible to users:

- For all Java clients this is the value of the `user.home` Java System Property.
- For C clients on UNIX and Linux platforms this is the value of the HOME environment variable.
- For C clients on Windows this is the concatenated values of the HOMEDRIVE and HOMEPATH environment variables.

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

Which IBM MQ clients can read each attribute

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

Table 106. Which attributes apply to each type of client

`mqclient.ini` stanza name and attributes	**Description**	**C and unmanaged .NET**	**Java**	**JMS**	**Managed .NET and XMS .NET**	
CHANNELS stanza						
CCSID	The coded character set number to be used.	Yes	No	No	Yes	
ChannelDefinitionDirectory	The directory path to the file containing the client channel definition table.	Yes	No	No	Yes	
ChannelDefinitionFile	The name of the file containing the client channel definition table.	Yes	No	No	No	
ReconDelay	An administrative option to configure reconnect delay for client programs that can auto-reconnect.	Yes	No	Yes	No	
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	No	

Table 106. Which attributes apply to each type of client (continued)

`mqclient.ini` stanza name and attributes	Description	C and unmanaged .NET	Java	JMS	Managed .NET and XMS .NET	
MQReconnectTimeout	The timeout in seconds to reconnect to a client.	Yes	No	No	No	
ServerConnectionParms	The location of the IBM MQ server and the communication method to be used.	Yes	No	No	Yes	
Put1DefaultAlwaysSync	Controls the behavior of the MQPUT1 function call with the option `MQPMO_RESPONSE_AS_Q_DEF`.	Yes	Yes	Yes	No	
PasswordProtection	Allows you to set protected passwords in the MQCSP structure, rather than using SSL or TLS.	Yes	Yes	Yes	No	
ClientExitPath stanza						
ExitsDefaultPath	Specifies the location of 32-bit channel exits for clients.	Yes	No	No	Yes	
ExitsDefaultPath64	Specifies the location of 64-bit channel exits for clients.	Yes	No	No	Yes	
JavaExitsClassPath	The values to be added to the classpath when a Java exit is run.	No	Yes	Yes	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	
MessageBuffer stanza						

Table 106. Which attributes apply to each type of client (continued)

`mqclient.ini` stanza name and attributes	Description	C and unmanaged .NET	Java	JMS	Managed .NET and XMS .NET	
MaximumSize	Size, in kilobytes, of the read-ahead buffer, in the range 1 through 999 999.	Yes	Yes	Yes	Yes	
PurgeTime	Interval, in seconds, after which messages left in the read-ahead buffer are purged.	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	
PreConnect stanza						
Data	URL of the repository where connection definitions are stored.	Yes	No	No	No	
Function	Name of the functional entry point into the library that contains the PreConnect exit code.	Yes	No	No	No	
Module	The name of the module containing the API exit code.	Yes	No	No	No	
Sequence	The sequence in which this exit is called relative to other exits.	Yes	No	No	No	
SSL stanza						

Table 106. Which attributes apply to each type of client (continued)

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

Table 106. Which attributes apply to each type of client (continued)

`mqclient.ini` stanza name and attributes	Description	C and unmanaged .NET	Java	JMS	Managed .NET and XMS .NET	
SSLCryptoHardware	Sets the parameter string required to configure PKCS #11 cryptographic hardware present on the system.	Yes	No	No	No	
SSLFipsRequired	Specifies whether only FIPS-certified algorithms are to be used if cryptography is carried out in IBM MQ.	Yes	No	No	No	
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	
SSLKeyRepository	The location of the key repository that holds the user's digital certificate, in stem format.	Yes	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	
TCP stanza						
ClntRcvBuffSize	The size in bytes of the TCP/IP receive buffer used by the client end of a client-connection server-connection channel.	Yes	Yes	Yes	Yes	

Table 106. Which attributes apply to each type of client (continued)

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

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

CHANNELS stanza of the client configuration file

Use the CHANNELS stanza to specify information about client channels.

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

The following attributes can be included in the CHANNELS stanza:

CCSID = *number*
The coded character set number to be used.

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

The CCSID number is equivalent to the MQCCSID environment parameter.

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

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

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

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

V 9.0.0 ChannelDefinitionDirectory can contain a URL which works in combination with the ChannelDefinitionFile attribute (see "Web addressable access to the client channel definition table" on page 865).

The ChannelDefinitionDirectory path is equivalent to the MQCHLLIB environment parameter.

ChannelDefinitionFile = *filename*|AMQCLCHL.TAB

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

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

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

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

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

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

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

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

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

NO Unless overridden by **MQCONNX**, the client is not reconnected automatically.

YES Unless overridden by **MQCONNX**, the client reconnects automatically.

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

DISABLED

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

The automatic client reconnection depends on two values:

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

Table 107. Automatic reconnection depends on the values set in the application and in the mqclient.ini file

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

`MQReconnectTimeout`

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

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

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).

`ServerConnectionParms`

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

This attribute can be read by C, unmanaged .NET, and managed .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| YES`

Controls the behavior of the MQPUT1 function call with the option `MQPMO_RESPONSE_AS_Q_DEF`.

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

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

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

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

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

See MQCSP password protection for further information.

Related information:
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, and managed .NET clients.

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

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

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

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

In the JMS configuration file, the JavaExitsClassPath name is given the standard com.ibm.mq.cfg. prefix and this full name is also used on the IBM WebSphere MQ Version 7.0 or later system property. At Version 6.0 this attribute was specified using system property com.ibm.mq.exitClasspath, which was documented in the Version 6.0 readme file. The use of com.ibm.mq.exitClasspath is deprecated. If both JavaExitsClassPath and exitClasspath are present, JavaExitsClassPath is honored. If only exitClasspath usage is present, it is still honored in IBM WebSphere MQ Version 7.0 or later.

JMQI stanza of the client configuration file

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

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

The following attribute can be included in the JMQI stanza:

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

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

This attribute can have the following values:

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

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

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

Windows

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

LU62 stanza

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

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

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

TPName
The TP name to start on the remote site.

NETBIOS stanza

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

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

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

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

This is equivalent to the MQNAME environment parameter.

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

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

SPX stanza

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

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

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

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

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

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

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

MessageBuffer stanza of the client configuration file

Use the MessageBuffer stanza to specify information about message buffers.

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

The following attributes can be included in the MessageBuffer stanza:

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

: This attribute can be read by C, unmanaged .NET, IBM MQ classes for Java, IBM MQ classes for JMS, and managed .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, and managed .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, and managed .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 = Upper - Lower
```

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

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

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

Example 1:

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

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

Using the calculation

```
T = Upper - Lower
```

T is (100 - 20) = 80 Kb.

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

Example 2:

The MaximumSize 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 = Upper - Lower
```

T is (100 - 40) = 60 Kb

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

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

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

PreConnect stanza of the client configuration file

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

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

The following attributes can be included in the PreConnect stanza:

`Data =` ***`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 information:
Referencing connection definitions using a pre-connect exit from a repository

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:

`CDPCheckExtensions = YES|NO`

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.

`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.

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

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

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

`EncryptionPolicySuiteB = *string*`

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

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

This attribute has the following possible values:

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

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

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

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

`OCSPAuthentication = OPTIONAL|REQUIRED|WARN`

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

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

This attribute has the following possible values:

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

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

  This value is the default value.

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

`OCSPCheckExtensions = 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`.

`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.

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

`SSLFipsRequired = YES|NO`

Specifies whether only FIPS-certified algorithms are to be used if cryptography is carried out in IBM MQ.

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

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

`SSLHTTPProxyName =` ***`string`***

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

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

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

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

`SSLKeyRepository =` ***`pathname`***

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

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

`SSLKeyResetCount =` ***`integer`*** **`|0`**

The number of unencrypted bytes sent and received on 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.

TCP stanza of the client configuration file

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

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

The following attributes can be included in the TCP stanza:

`ClntRcvBuffSize = `*`number`*`|0`

The size in bytes of the TCP/IP receive buffer used by the client end of a client-connection server-connection channel.

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

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

`ClntSndBuffSize = `*`number`*`|0`

The size in bytes of the TCP/IP send buffer used by the client end of a client-connection server-connection channel.

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

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

`Connect_Timeout = `*`number`*

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

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

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

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

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

`IPAddressVersion = MQIPADDR_IPV4|MQIPADDR_IPV6`

Specifies which IP protocol to use for a channel connection.

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

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

KeepAlive = YES|NO
Switch the KeepAlive function on or off. KeepAlive=YES causes TCP/IP to check periodically that the other end of the connection is still available. If it is not, the channel is closed.

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

Windows **Library1 = *DLLName*|WSOCK32**
(Windows only) The name of the TCP/IP sockets DLL.

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

TMF and TmfGateway stanzas

NSS Client

Use the TMF and TMF/Gateway stanzas to specify the required configuration parameters for the IBM MQ client for HP Integrity NonStop Server to communicate with the TMF/Gateway.

If you want to use TMF, then you must define a TMF stanza and one TmfGateway stanza for each queue manager with which you are communicating. All values are derived from your configuration.

The IBM MQ provided TMF/Gateway runs in a Pathway environment.

TMF stanza

PathMon = *name*
The name of your defined Pathmon process that defines the server classes for the TMF/Gateway.

TmfGateway stanza

The following attributes can be included in this stanza:

QManager = *name*
The name of the queue manager.

Server = *name*
The server class name for the TMF/Gateway configured for that queue manager.

Example

Here is an example of a TMF stanza that is defined with two TmfGateway stanzas for two different queue managers on different servers:

```
TMF:
PathMon=$PSD1P

TmfGateway:
QManager=MQ5B
Server=MQ-MQ5B

TmfGateway:
QManager=MQ5C
Server=MQ-MQ5C
```

Related concepts:

"Gateway process overview" on page 1413
The HP NonStop Transaction Management Facility (TMF) provides services to enable a gateway process to register as a resource manager. The TMF/Gateway process used by IBM MQ client for HP Integrity NonStop Server runs under Pathway.

"Configuring the client initialization file" on page 1415
If you are using the HP NonStop Transaction Management Facility (TMF), you must have an IBM MQ client initialization file to enable your IBM MQ client for HP Integrity NonStop Server to reach the TMF Gateway.

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 variable. These commands are available on all the IBM MQ MQI client platforms unless otherwise stated.

IBM MQ uses default values for those variables that you have not set.

Note: z/OS IBM MQ for z/OS does not support any IBM MQ environment variables. If you are using this platform as your server, see Client channel definition table for information about how the client channel definition table is generated on z/OS. You can still use the IBM MQ environment variables on your client platform.

Procedure

- Windows 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 command `SET MQSERVER=`
  - To display the current setting of an environment variable, use the command `SET MQSERVER`
  - To display all environment variables for the session, use the command `set`
- UNIX Linux On UNIX 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 command `unset MQSERVER`.
  - To display the current setting of an environment variable, use the command `echo $MQSERVER`.
  - To display all environment variables for the session, use the command `set`.

Related tasks:

"Configuring a client using a configuration file" on page 876
You 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 information:

Environment variables

MQCCDTURL

V 9.0.0

MQCCDTURL provides the equivalent capability to setting a combination of the MQCHLLIB and MQCHLTAB environment variables.

Attention: You can use the environment variable option 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.

However MQCCDTURL only accepts a URL value; MQCCDTURL will not accept the existing local file system directory format.

To use MQCCDTURL, in place of MQCHLLIB and MQCHLTAB, to use a local file, you can use a `'file://'` protocol. Therefore:

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

is equivalent to:

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

Related concepts:

V 9.0.0 "Web addressable access to the client channel definition table" on page 865
From Version 9.0, IBM MQ provides the ability to locate a client channel definition table (CCDT) through a URL, either by programming using MQCNO, using environment variables, or using `mqclient.ini` file stanzas.

"Client channel definition table" on page 863
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 CCDT is created automatically. You must then make it available to the client application.

Related reference:

"MQCHLLIB" on page 899
`MQCHLLIB` specifies the directory path to the file containing the client channel definition table (CCDT). The file is created on the server, but can be copied across to the IBM MQ MQI client workstation.

"MQCHLTAB" on page 901
MQCHLTAB specifies the name of the file containing the client channel definition table (ccdt). The default file name is AMQCLCHL.TAB.

Related information:

CCDTURL

XMSC_WMQ_CCDTURL

Connecting IBM MQ MQI applications to queue managers

MQCCSID

MQCCSID specifies the coded character set number to be used and overrides the CCSID value with which the server has been configured.

See Choosing client or server coded character set identifier (CCSID) for more information.

To set this variable use one of the following commands:

- Windows On Windows:

  ```
  SET MQCCSID=number
  ```

- UNIX Linux On UNIX and Linux:

  ```
  export MQCCSID=number
  ```

- IBM i On IBM i:

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

MQCERTLABL

MQCERTLABL specifies the certificate label of the channel definition.

See Certificate label (CERTLABL) for more information.

MQCERTVPOL

MQCERTVPOL specifies the certificate validation policy used.

For more information about certificate validation policies in IBM MQ, see Certificate validation policies in IBM MQ.

This environment variable overrides the *CertificateValPolicy* setting in the SSL stanza of the client ini file. The variable can be set to one of two values:

ANY Use any certificate validation policy supported by the underlying secure sockets library.

RFC5280
Use only certificate validation which complies with the RFC 5280 standard.

To set this variable, use one of these commands:

- Windows On Windows:

  ```
  SET MQCERTVPOL= value
  ```

- UNIX Linux For UNIX and Linux systems:

  ```
  export MQCERTVPOL= value
  ```

- IBM i For IBM i:

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

MQCHLLIB

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

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

- Windows On Windows: *MQ_INSTALLATION_PATH*
- UNIX Linux On UNIX and Linux: /var/mqm/
- 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.

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

Or:

- Windows On Windows: *MQ_INSTALLATION_PATH*\data\qmgrs\qmgrname\@ipcc
- UNIX Linux On UNIX and Linux: /prefix/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:
  - UNIX Linux /var/mqm on UNIX 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:
  - IBM i UNIX Linux In the mqs.ini file on IBM i, UNIX, and Linux.
  - Windows In the registry on Windows.

Notes:

1. z/OS If you are using IBM MQ for z/OS as your server, the file must be kept on the IBM MQ client workstation.
2. If set, MQCHLLIB overrides the path used to locate the CCDT.
3. V 9.0.0 MQCHLLIB can contain a URL which works in combination with the MQCHLTAB environment variable (see "Web addressable access to the client channel definition table" on page 865).
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 quotations marks.

Examples

To set this variable use one of these commands:

- Windows On Windows:
```
SET MQCHLLIB=pathname
```
For example:
```
SET MQCHLLIB=C:\wmqtest
```
- UNIX Linux For UNIX and Linux systems:
```
export MQCHLLIB=pathname
```
- IBM i For IBM i:
```
ADDENVVAR ENVVAR(MQCHLLIB) VALUE(pathname)
```

Related reference:

"MQCHLTAB"
MQCHLTAB specifies the name of the file containing the client channel definition table (ccdt). The default file name is AMQCLCHL.TAB.

Related information:

Connecting IBM MQ MQI applications to queue managers

MQCHLTAB

MQCHLTAB specifies the name of the file containing the client channel definition table (ccdt). The default file name is AMQCLCHL.TAB.

For information about where the client channel definition table is located on a server, see "Client channel definition table" on page 863.

To set this variable use one of these commands:

- Windows On Windows:
```
SET MQCHLTAB=filename
```
- UNIX Linux On UNIX and Linux:
```
export MQCHLTAB=filename
```
- IBM i On IBM i:
```
ADDENVVAR ENVVAR(MQCHLTAB) VALUE(filename)
```

For example:
```
SET MQCHLTAB=ccdf1.tab
```

In the same way as for the client, the MQCHLTAB environment variable on the server specifies the name of the client channel definition table.

Related concepts:

V 9.0.0 "Web addressable access to the client channel definition table" on page 865
From Version 9.0, IBM MQ provides the ability to locate a client channel definition table (CCDT) through a URL, either by programming using MQCNO, using environment variables, or using `mqclient.ini` file stanzas.

Related reference:

"MQCHLLIB" on page 899
`MQCHLLIB` specifies the directory path to the file containing the client channel definition table (CCDT). The file is created on the server, but can be copied across to the IBM MQ MQI client workstation.

V 9.0.0 "MQCCDTURL" on page 897
MQCCDTURL provides the equivalent capability to setting a combination of the MQCHLLIB and MQCHLTAB environment variables.

Related information:

Connecting IBM MQ MQI applications to queue managers

MQIPADDRV

MQIPADDRV specifies which IP protocol to use for a channel connection. It has the possible string values of "MQIPADDR_IPv4" or "MQIPADDR_IPv6". These values have the same meanings as IPv4 and IPv6 in ALTER QMGR IPADDRV. If it is not set, "MQIPADDR_IPv4" is assumed.

To set this variable use one of these commands:

- Windows On Windows:

```
SET MQIPADDRV=MQIPADDR_IPv4|MQIPADDR_IPv6
```

- UNIX Linux On UNIX and Linux:

```
export MQIPADDRV=MQIPADDR_IPv4|MQIPADDR_IPv6
```

- IBM i On IBM i:

```
ADDENVVAR ENVVAR(MQIPADDRV) VALUE(MQIPADDR_IPv4|MQIPADDR_IPv6)
```

MQNAME

MQNAME specifies the local NetBIOS name that the IBM MQ processes can use.

See "Defining a NetBIOS connection on Windows" on page 1030 for a full description and for the rules of precedence on the client and the server.

To set this variable use this command:

```
SET MQNAME=Your_env_Name
```

For example:

```
SET MQNAME=CLIENT1
```

The NetBIOS on some platforms requires a different name (set by MQNAME) for each application if you are running multiple IBM MQ applications simultaneously on the IBM MQ MQI client.

MQSERVER

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.

You cannot use MQSERVER to define a TLS channel or a channel with channel exits. For details of how to define a TLS channel, see Protecting channels with TLS.

ConnectionName must be a fully-qualified network name. The *ChannelName* cannot contain the forward slash (/) character because this character is used to separate the channel name, transport type, and connection name. When the MQSERVER environment variable is used to define a client channel, a maximum message length (MAXMSGL) of 100 MB is used. Therefore the maximum message size in effect for the channel is the value specified in the SVRCONN channel at the server.

To set this variable use one of these commands:

- Windows On Windows:

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

- UNIX Linux On UNIX and Linux:

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

- IBM i On IBM i:

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

- z/OS On z/OS

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

TransportType can be one of the following values, depending on your IBM MQ client platform:

- LU62
- TCP
- NETBIOS
- SPX

ConnectionName can be a comma-separated list of connection names. The connection names in the list are used in a similar way to multiple connections in a client connection table. The *ConnectionName* list might be used as an alternative to queue manager groups to specify multiple connections for the client to try. If you are configuring a multi-instance queue manager, you might use a *ConnectionName* list to specify different queue manager instances.

TCP/IP default port:

By default, for TCP/IP, IBM MQ assumes that the channel will be connected to port 1414.

You can change this by:
- Adding the port number in brackets as the last part of the ConnectionName:
  - Windows On Windows:
    ```
    SET  MQSERVER=ChannelName/TransportType/ConnectionName(PortNumber)
    ```
  - UNIX Linux On UNIX and Linux:
    ```
    export  MQSERVER='ChannelName/TransportType/ConnectionName(PortNumber)'
    ```
- Changing the `mqclient.ini` file by adding the port number to the protocol name, for example:
  ```
  TCP:
  port=2001
  ```
- Adding IBM MQ to the services file as described in "Using the TCP/IP listener on UNIX and Linux" on page 1035.

SPX default socket:

By default, for SPX, IBM MQ assumes that the channel will be connected to socket 5E86.

You can change this by:
- Adding the socket number in brackets as the last part of the ConnectionName:
  ```
  SET  MQSERVER=ChannelName/TransportType/ConnectionName(SocketNumber)
  ```
  For SPX connections, specify the ConnectionName and socket in the form `network.node(socket)`. If the IBM MQ client and server are on the same network, the network need not be specified. If you are using the default socket, the socket need not be specified.
- Changing the `qm.ini` file by adding the port number to the protocol name, for example:
  ```
  SPX:
  socket=5E87
  ```

Using MQSERVER:

If you use the MQSERVER environment variable to define the channel between your IBM MQ MQI client machine and a server machine, this is the only channel available to your application, and no reference is made to the client channel definition table (CCDT).

In this situation, the listener program that you have running on the server machine determines the queue manager to which your application will connect. It will be the same queue manager as the listener program is connected to.

If the MQCONN or MQCONNX request specifies a queue manager other than the one the listener is connected to, or if the MQSERVER parameter *TransportType* is not recognized, the MQCONN or MQCONNX request fails with return code MQRC_Q_MGR_NAME_ERROR.

UNIX Linux On UNIX and Linux systems, you might define MQSERVER as in one of the following examples:

```
export MQSERVER=CHANNEL1/TCP/'9.20.4.56(2002)'
export MQSERVER=CHANNEL1/LU62/BOX99
```

All MQCONN or MQCONNX requests then attempt to use the channel you have defined unless an MQCD structure has been referenced from the MQCNO structure supplied to MQCONNX, in which case the channel specified by the MQCD structure takes priority over any specified by the MQSERVER environment variable.

The MQSERVER environment variable takes priority over any client channel definition pointed to by MQCHLLIB and MQCHLTAB.

Canceling MQSERVER

To cancel MQSERVER and return to the client channel definition table pointed to by MQCHLLIB and MQCHLTAB, enter the following:

- Windows On Windows:
  ```
  SET MQSERVER=
  ```
- UNIX Linux On UNIX and Linux:
  ```
  unset MQSERVER
  ```

MQSSLCRYP

MQSSLCRYP holds a parameter string that allows you to configure the cryptographic hardware present on the system. The permitted values are the same as for the SSLCRYP parameter of the ALTER QMGR command.

To set this variable use one of these commands:

- Windows On Windows systems:
  ```
  SET MQSSLCRYP=string
  ```
- UNIX Linux On UNIX and Linux systems:
  ```
  export MQSSLCRYP=string
  ```

MQSSLFIPS

MQSSLFIPS specifies whether only FIPS-certified algorithms are to be used if cryptography is carried out in IBM MQ. The values are the same as for the SSLFIPS parameter of the ALTER QMGR command.

The use of FIPS-certified algorithms is affected by the use of cryptographic hardware, see Specifying that only FIPS-certified CipherSpecs are used at run time on the MQI client.

To set this variable use one of these commands:

- Windows On Windows systems:
  ```
  SET MQSSLFIPS=YES|NO
  ```
- UNIX Linux On UNIX and Linux systems:
  ```
  export MQSSLFIPS=YES|NO
  ```
- IBM i On IBM i:
  ```
  ADDENVVAR ENVVAR(MQSSLFIPS) VALUE(YES|NO)
  ```

The default is `NO`.

MQSSLKEYR

MQSSLKEYR specifies the location of the key repository that holds the digital certificate belonging to the user, in stem format. Stem format means that it includes the full path and the file name without an extension.

For full details, see the SSLKEYR parameter of the ALTER QMGR command.

To set this variable use one of these commands:

- Windows On Windows systems:
  ```
  SET MQSSLKEYR=pathname
  ```
- UNIX Linux On UNIX and Linux systems:
  ```
  export MQSSLKEYR=pathname
  ```
- IBM i On IBM i:
  ```
  ADDENVVAR ENVVAR(MQSSLKEYR) VALUE(pathname)
  ```

There is no default value.

MQSSLPROXY

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

To set this variable use one of these commands:

- Windows On Windows systems:
  ```
  SET MQSSLPROXY= string
  ```
- UNIX Linux On UNIX and Linux systems:
  ```
  export MQSSLPROXY="string"
  ```

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

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

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

MQSSLRESET

MQSSLRESET represents the number of unencrypted bytes sent and received on a TLS channel before the secret key is renegotiated.

See Resetting TLS secret keys for more information about secret key renegotiation.

It can be set to an integer in the range 0 through 999 999 999. The default is 0, which indicates that secret keys are never renegotiated. If you specify 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 this variable use one of these commands:

- Windows On Windows systems:
  ```
  SET MQSSLRESET=integer
  ```
- UNIX Linux On UNIX and Linux systems:
  ```
  export MQSSLRESET=integer
  ```
- IBM i On IBM i:
  ```
  ADDENVVAR ENVVAR(MQSSLRESET) VALUE(integer)
  ```

MQSUITEB

ULW

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

Suite B restricts the set of enabled cryptographic algorithms in order to provide an assured level of security.

See Configuring IBM MQ for Suite B for more information.

MQTCPTIMEOUT

How long IBM MQ waits for a TCP connect call.

Changing IBM MQ and queue manager configuration information

You can change the behavior of IBM MQ or an individual queue manager to suit the needs of your installation.

About this task

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

You change attribute information by editing the IBM MQ configuration files. You can edit configuration files either automatically by using commands that change the configuration of queue managers on the node or manually by using a standard text editor. For more information, see “Editing configuration files” on page 908.

Linux Windows On Windows and Linux (x86 and x86-64 platforms), you can also edit the IBM MQ configuration files by using the IBM MQ Explorer.

Windows On Windows systems you can also use **amqmdain** to change configuration information, as described in amqmdain.

Procedure

For more information about configuring IBM MQ and queue managers for your platform, see the following subtopics:.

Related concepts:

"Configuring IBM MQ" on page 833
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.

Related tasks:

z/OS "Configuring queue managers on z/OS" on page 1455
Use these instructions to configure queue managers on IBM MQ for z/OS.

Related information:

Planning

Administering IBM MQ

Changing configuration information on UNIX, Linux, and Windows

ULW

On UNIX, Linux, and Windows, you can change the IBM MQ configuration attributes that are held in configuration files, at the level of the node and of the queue manager.

About this task

On UNIX, Linux, and Windows platforms, you can change IBM MQ configuration attributes within the following files:

- An IBM MQ configuration file (`mqs.ini`) to effect changes for IBM MQ on the node as a whole. There is one `mqs.ini` file for each node. For more information on the stanzas included in `mqs.ini`, see "Attributes for changing IBM MQ configuration information" on page 927.
- A queue manager configuration file (`qm.ini`) to effect changes for specific queue managers. There is one `qm.ini` file for each queue manager on the node. For more information on the stanzas included in `qm.ini`, see "Changing queue manager configuration information" on page 934.

Client configuration options are held separately, in the client configuration file, `qm.ini`. For more information, see "Configuring a client using a configuration file" on page 876.

A configuration file (or stanza file) contains one or more stanzas, which are groups of lines in the `.ini` file that together have a common function or define part of a system, such as log functions, channel functions, and installable services.

Because the IBM MQ configuration file is used to locate the data associated with queue managers, a nonexistent or incorrect configuration file can cause some or all MQSC commands to fail. Also, applications cannot connect to a queue manager that is not defined in the IBM MQ configuration file.

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

Procedure

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

- Edit the configuration files using commands or a standard text editor. For more information, see "Editing configuration files."
- Linux Windows On Windows and Linux (x86 and x86-64) systems use IBM MQ Explorer to make changes to the configuration files. For more information, see Configuring IBM MQ using MQ Explorer.
- Windows On Windows systems, as an alternative to using IBM MQ Explorer, use the **amqmdain** command to make changes to the configuration files. For more information, see amqmdain.

Related concepts:

"Configuring IBM MQ" on page 833
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.

Related tasks:

"Changing IBM MQ and queue manager configuration information" on page 906
You can change the behavior of IBM MQ or an individual queue manager to suit the needs of your installation.

IBM i "Changing configuration information on IBM i" on page 914
You can change the behavior of queue managers to suit your installation's needs by modifying the values specified on a set of configuration attributes (or parameters) that govern IBM MQ.

"Changing queue manager configuration information" on page 934
The attributes that you can use to modify the configuration of an individual queue manager override any settings for IBM MQ.

Related reference:

"Attributes for changing IBM MQ configuration information" on page 927
On IBM MQ for Windows systems and on IBM MQ for Linux (x86 and x86-64 platforms) systems, modify configuration information using the IBM MQ Explorer. On other systems, modify the information by editing the mqs.ini configuration file.

Related information:

Planning

Administering IBM MQ

Editing configuration files

ULW

Edit configuration files using commands or a standard text editor.

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

You can edit configuration files either:
- Automatically, using commands that change the configuration of queue managers on the node
- Manually, using a standard text editor

You can edit the default values in the IBM MQ configuration files after installation.

If you set an incorrect value on a configuration file attribute, the value is ignored and an operator message is issued to indicate the problem. (The effect is the same as missing out the attribute entirely.)

When you create a new queue manager:
- Back up the IBM MQ configuration file
- Back up the new queue manager configuration file

Comments can be included in configuration files by adding a ";" or a "#" character before the comment text. If you want to use a ";" or a "#" character without it representing a comment, you can prefix the character with a "\" character and it will be used as part of the configuration data.

When do you need to edit a configuration file?

Edit a configuration file to recover from backup, move a queue manager, change the default queue manager or to assist IBM support.

You might need to edit a configuration file if, for example:
- You lose a configuration file. (Recover from backup if you can.)
- You need to move one or more queue managers to a new directory.
- You need to change your default queue manager; this could happen if you accidentally delete the existing queue manager.
- You are advised to do so by your IBM Support Center.

Configuration file priorities

The value of an attribute is defined in multiple places. Attributes set in commands take precedence over attributes in configuration files.

The attribute values of a configuration file are set according to the following priorities:
- Parameters entered on the command line take precedence over values defined in the configuration files
- Values defined in the qm.ini files take precedence over values defined in the mqs.ini file

IBM MQ configuration file, mqs.ini

ULW

The IBM MQ configuration file, `mqs.ini`, contains information relevant to all the queue managers on the node. It is created automatically during installation.

Directory locations

Linux UNIX On UNIX 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 UNIX 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 913).

Windows The `mqs.ini` file for Windows systems is given by the WorkPath specified in the `HKLM\SOFTWARE\IBM\WebSphere MQ` key. It contains:
- The names of the queue managers
- The name of the default queue manager
- The location of the files associated with each of them

`LogDefaults` stanza for a new installation

The supplied `LogDefaults` stanza for a new IBM MQ installation does not contain any explicit values for the attributes. The lack of an attribute means that the default for this value is used upon creation of a new queue manager. The default values are shown for the `LogDefaults` stanza in Figure 80 on page 910. A value of zero for the `LogBufferPages` attribute means 512.

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

Example `mqs.ini` file

UNIX

```
#*******************************************************************#
#* Module Name: mqs.ini                                            *#
#* Type       : IBM MQ Machine-wide Configuration File             *#
#* Function   : Define IBM MQ resources for an entire machine      *#
#*******************************************************************#
#* Notes      :                                                    *#
#* 1) This is the installation time default configuration          *#
#*                                                                 *#
#*******************************************************************#
AllQueueManagers:
#*******************************************************************#
#* The path to the qmgrs directory, below which queue manager data *#
#* is stored                                                       *#
#*******************************************************************#
DefaultPrefix=/var/mqm

LogDefaults:
   LogPrimaryFiles=3
   LogSecondaryFiles=2
   LogFilePages=4096
   LogType=CIRCULAR
   LogBufferPages=0
   LogDefaultPath=/var/mqm/log

QueueManager:
   Name=saturn.queue.manager
   Prefix=/var/mqm
   Directory=saturn!queue!manager
   InstallationName=Installation1

QueueManager:
   Name=pluto.queue.manager
   Prefix=/var/mqm
   Directory=pluto!queue!manager
   InstallationName=Installation2

DefaultQueueManager:
   Name=saturn.queue.manager

ApiExitTemplate:
   Name=OurPayrollQueueAuditor
   Sequence=2
   Function=EntryPoint
   Module=/usr/ABC/auditor
   Data=123

ApiExitCommon:
   Name=MQPoliceman
   Sequence=1
   Function=EntryPoint
   Module=/usr/MQPolice/tmqp
   Data=CheckEverything
```

Figure 80. Example of an IBM MQ configuration file for UNIX

Queue manager configuration files, qm.ini

ULW

A queue manager configuration file, `qm.ini`, contains information relevant to a specific queue manager.

There is one queue manager configuration file for each queue manager. The `qm.ini` file is automatically created when the queue manager with which it is associated is created.

Note: For information on when any changes that you make to the `qm.ini` file take effect, see "Changing configuration information on UNIX, Linux, and Windows" on page 907

V 9.0.4 V 9.0.0.2 From IBM MQ Version 9.0.4 and IBM MQ Version 9.0.0, Fix Pack 2, 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 UNIX On UNIX and Linux systems, a `qm.ini` file is held in the root of the directory tree occupied by the queue manager. For example, the path and the name for a configuration file for a queue manager called QMNAME is:

```
/var/mqm/qmgrs/QMNAME/qm.ini
```

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
```

The queue manager name can be up to 48 characters in length. However, this does not guarantee that the name is valid or unique. Therefore, a directory name is generated based on the queue manager name. This process is known as *name transformation*. For a description, see Understanding IBM MQ file names.

Linux UNIX

Example `qm.ini` file

Figure 81 on page 912 shows how groups of attributes might be arranged in a queue manager configuration file in IBM MQ for UNIX and Linux systems.

From IBM MQ Version 9.0.5, this is an example queue manager configuration file:

```
#* Module Name: qm.ini                                             *#
#* Type       : IBM MQ queue manager configuration file            *#
#  Function   : Define the configuration of a single queue manager *#
#*                                                                 *#
#*******************************************************************#
#* Notes      :                                                    *#
#* 1) This file defines the configuration of the queue manager     *#
#*                                                                 *#
#*******************************************************************#

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

Service:
   Name=AuthorizationService
   EntryPoints=14

ServiceComponent:
   Service=AuthorizationService
   Name=MQSeries.UNIX.auth.service
   Module=amqzfu
   ComponentDataSize=0

Log:
   LogPrimaryFiles=3
   LogSecondaryFiles=2
   LogFilePages=4096
   LogType=CIRCULAR
   LogBufferPages=0 1
   LogPath=/var/mqm/log/saturn!queue!manager/

XAResourceManager:
   Name=DB2 Resource Manager Bank
   SwitchFile=/usr/bin/db2swit
   XAOpenString=MQBankDB
   XACloseString=
   ThreadOfControl=THREAD

Channels: 2
   MaxChannels=200
   MaxActiveChannels=100
   MQIBindType=STANDARD

TCP:
SndBuffSize=0
RcvBuffSize=0
RcvSndBuffSize=0
RcvRcvBuffSize=0
ClntSndBuffSize=0
ClntRcvBuffSize=0
SvrSndBuffSize=0
SvrRcvBuffSize=0

QMErrorLog:
   ErrorLogSize=262144
   ExcludeMessage=7234
   SuppressMessage=9001,9002,9202
   SuppressInterval=30

ApiExitLocal:
   Name=ClientApplicationAPIchecker
   Sequence=3
   Function=EntryPoint
   Module=/usr/Dev/ClientAppChecker
   Data=9.20.176.20

TuningParameters:
   ImplSyncOpenOutput=2
```

For IBM MQ Version 9.0.4 and earlier, and LTS, this is an example configuration file:

Notes for Figure 81 on page 912:

1. The value of zero for `LogBufferPages` gives a value of 512.
2. For more information on the Channel stanza, see “Initialization and configuration files” on page 1011.
3. The maximum number of XAResourceManager stanzas is limited to 255. However, you should use only a small number of stanzas to avoid transaction performance degradation.

AccessMode stanza

The `qm.ini` file for Windows includes an additional *AccessMode* stanza:

```
AccessMode:
SecurityGroup=wmq\wmq
```

`APIExitLocal` stanza

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.

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
```

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.

Installation configuration file, mqinst.ini

On UNIX or Linux, the installation configuration file, `mqinst.ini`, contains information about all the IBM MQ installations. On Windows, installation configuration information is in the registry.

Location of the `mqinst.ini` file

The `mqinst.ini` file is in the `/etc/opt/mqm` directory on UNIX and Linux systems. It contains information about which installation, if any, is the primary installation as well as the following information for each installation:

- The installation name
- The installation description
- The installation identifier

- The installation path

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.

dltmqinst to delete entries.

dspmqinst to display entries.

setmqinst to set entries.

Windows

Installation configuration information on 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.

Windows

Changing configuration information on IBM i

IBM i

You can change the behavior of queue managers to suit your installation's needs by modifying the values specified on a set of configuration attributes (or parameters) that govern IBM MQ.

About this task

You change configuration attributes by editing the IBM MQ configuration files.

Procedure

For information on modifying the configuration values on IBM i, see the following topics:
- “IBM MQ configuration files for IBM i” on page 915
- “Attributes for changing configuration information on IBM i” on page 917
- “Changing queue manager configuration information on IBM i” on page 919
- “Example mqs.ini and qm.ini files for IBM i” on page 925

Related concepts:

"Configuring IBM MQ" on page 833
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.

Related tasks:

"Changing IBM MQ and queue manager configuration information" on page 906
You can change the behavior of IBM MQ or an individual queue manager to suit the needs of your installation.

"Changing configuration information on UNIX, Linux, and Windows" on page 907
On UNIX, Linux, and Windows, you can change the IBM MQ configuration attributes that are held in configuration files, at the level of the node and of the queue manager.

"Changing queue manager configuration information" on page 934
The attributes that you can use to modify the configuration of an individual queue manager override any settings for IBM MQ.

Related reference:

"Attributes for changing IBM MQ configuration information" on page 927
On IBM MQ for Windows systems and on IBM MQ for Linux (x86 and x86-64 platforms) systems, modify configuration information using the IBM MQ Explorer. On other systems, modify the information by editing the mqs.ini configuration file.

Related information:

Planning

Administering IBM MQ

IBM MQ configuration files for IBM i

▶ IBM i

Use this information to understand the methods for configuring IBM MQ for IBM i.

On IBM i, you modify IBM MQ configuration attributes within:
- An IBM MQ configuration file, `mqs.ini`, effect changes on the node as a whole. There is one `mqs.ini` file for each IBM MQ installation.
- A queue manager configuration file, `qm.ini`, effect changes for specific queue managers. There is one `qm.ini` file for each queue manager on the node.

Note that .ini files are stream files resident in the IFS.

A configuration file (which can be referred to as a ***stanza*** file) contains one or more stanzas, which are groups of lines in the .ini file that together have a common function or define part of a system, for example, log functions and channel functions. Any changes made to a configuration file do not take effect until the next time the queue manager is started.

Editing configuration files

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

You can edit configuration files either:
- Automatically, using commands that change the configuration of queue managers on the node.
- Manually, using the EDTF CL editor.

You can edit the default values in the IBM MQ configuration files after installation. If you set an incorrect value on a configuration file attribute, the value is ignored and an operator message is issued to indicate the problem. (The effect is the same as missing out the attribute entirely.)

When you create a new queue manager:
- Back up the IBM MQ configuration file.
- Back up the new queue manager configuration file.

When do you need to edit a configuration file?

You might need to edit a configuration file if, for example:
- You lose a configuration file; recover from backup if possible.
- You need to move one or more queue managers to a new directory.
- You need to change your default queue manager; this could happen if you accidentally delete the existing queue manager.
- You are advised to do so by your IBM Support Center.

Configuration file priorities

The attribute values of a configuration file are set according to the following priorities:
- Parameters entered on the command line take precedence over values defined in the configuration files.
- Values defined in the `qm.ini` files take precedence over values defined in the `mqs.ini` file.

The IBM MQ configuration file mqs.ini

The IBM MQ configuration file, `mqs.ini`, contains information relevant to all the queue managers on an IBM MQ installation. It is created automatically during installation. In particular, the `mqs.ini` file is used to locate the data associated with each queue manager.

The `mqs.ini` file is stored in `/QIBM/UserData/mqm`

The `mqs.ini` file contains:
- The names of the queue managers.
- The name of the default queue manager.
- The location of the files associated with each queue manager.
- Information identifying any API exits (see Configuring API exits for more information).

Queue manager configuration files, qm.ini

A queue manager configuration file, `qm.ini`, contains information relevant to a specific queue manager. There is one queue manager configuration file for each queue manager. The `qm.ini` file is automatically created when the queue manager with which it is associated is created.

A `qm.ini` file is held in the *`mqmdata directory`*`/`*`QMNAME`*`/qm.ini`, where *`mqmdata directory`* is `/QIBM/UserData/mqm` by default and *`QMNAME`* is the name of the queue manager to which the initialization file applies.

Note:
1. You can change the *`mqmdata directory`* in the `mqs.ini` file.
2. The queue manager name can be up to 48 characters in length. However, this does not guarantee that the name is valid or unique. Therefore, a directory name is generated based on the queue manager name. This process is known as **name transformation**. See Understanding IBM MQ for IBM i queue manager library names for further information.

Attributes for changing configuration information on IBM i

IBM i

Use this information to understand the configuration information stanzas.

The following groups of attributes occur in `mqs.ini`:
- "The AllQueueManagers stanza"
- "The DefaultQueueManager stanza" on page 918
- "The ExitProperties stanza" on page 918
- "The QueueManager stanza" on page 919

There are also two stanzas associated with API exits, ApiExitCommon and ApiExitTemplate. For details on using these, see Configuring API exits.

The AllQueueManagers stanza

The `AllQueueManagers` stanza can specify:
- The path to the `qmgrs` directory where the files associated with a queue manager are stored
- The path to the executable library
- The method for converting EBCDIC-format data to ASCII format

In the descriptions of the stanzas, the value underlined is the default value and the | symbol means *or*.

`DefaultPrefix=` ***`directory_name`***

The path to the `qmgrs` directory, within which the queue manager data is kept. If you change the default prefix for the queue manager, you must replicate the directory structure that was created at installation time. In particular, you must create the `qmgrs` structure. Stop IBM MQ before changing the default prefix, and restart IBM MQ only after moving the structures to the new location and changing the default prefix.

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

`ConvEBCDICNewline=NL_TO_LF|TABLE|ISO`

EBCDIC code pages contain a newline (NL) character that is not supported by ASCII code pages, although some ISO variants of ASCII contain an equivalent.

Use the ConvEBCDICNewline attribute to specify the method IBM MQ is to use when converting the EBCDIC NL character into ASCII format.

`NL_TO_LF`

Convert the EBCDIC NL character (X'15') to the ASCII line feed character, LF (X'0A'), for all EBCDIC to ASCII conversions.

NL_TO_LF is the default.

`TABLE`

Convert the EBCDIC NL character according to the conversion tables used on IBM i for all EBCDIC to ASCII conversions.

Note that the effect of this type of conversion can vary from language to language .

`ISO`

Specify ISO if you want:
- ISO CCSIDs to be converted using the TABLE method
- All other CCSIDs to be converted using the NL_TO_CF method.

Possible ISO CCSIDs are shown in Table 108 on page 918.

Table 108. List of possible ISO CCSIDs

CCSID	Code Set
819	ISO8859-1
912	ISO8859-2
915	ISO8859-5
1089	ISO8859-6
813	ISO8859-7
916	ISO8859-8
920	ISO8859-9
1051	roman8

If the ASCII CCSID is not an ISO subset, ConvEBCDICNewline defaults to NL_TO_LF.

The DefaultQueueManager stanza

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

`Name=` ***`default_queue_manager`***

The default queue manager processes any commands for which a queue manager name is not explicitly specified. The `DefaultQueueManager` attribute is automatically updated if you create a new default queue manager. If you inadvertently create a new default queue manager and then want to revert to the original, you must alter the `DefaultQueueManager` attribute manually.

The ExitProperties stanza

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

In the descriptions of the stanzas, the value underlined is the default value and the `|` symbol means *or*.

`CLWLMode= SAFE|FAST`

The cluster workload exit, CLWL, allows you to specify which cluster queue in the cluster is to be opened in response to an MQI call (for example: MQOPEN or MQPUT). The CLWL exit runs either in FAST mode or SAFE mode depending on the value you specify on the CLWLMode attribute. If you omit the CLWLMode attribute, the cluster workload exit runs in SAFE mode.

`SAFE`

Run the CLWL exit in a separate process to the queue manager. This is the default.

If a problem arises with the user-written CLWL exit when running in SAFE mode, the following happens:

- The CLWL server process (amqzlwa0) fails
- The queue manager restarts the CLWL server process
- The error is reported to you in the error log. If an MQI call is in progress, you receive notification in the form of a bad return code.

The integrity of the queue manager is preserved.

Note: Running the CLWL exit in a separate process might have a detrimental effect on performance.

`FAST`

Run the cluster exit inline in the queue manager process.

Specifying this option improves performance by avoiding the overheads associated with running in SAFE mode, but does so at the expense of queue manager integrity. Run the CLWL exit in

FAST mode only if you are convinced that there are **no** problems with your CLWL exit, and you are particularly concerned about performance overheads.

If a problem arises when the CLWL exit is running in FAST mode, the queue manager fails and you run the risk of compromising the integrity of the queue manager.

The QueueManager stanza

There is one `QueueManager` stanza for every queue manager. These attributes specify the queue manager name and the name of the directory containing the files associated with that queue manager. The name of the directory is based on the queue manager name, but is transformed if the queue manager name is not a valid file name.

See Understanding IBM MQ for IBM i queue manager library names for more information about name transformation.

`Name=` ***`queue_manager_name`***
The name of the queue manager.

`Prefix=` ***`prefix`***
Where the queue manager files are stored. By default, this is the same as the value specified on the DefaultPrefix attribute of the `AllQueueManager` stanza in the `mqs.ini` file.

`Directory=` ***`name`***
The name of the subdirectory under the *`prefix`*`\QMGRS` directory where the queue manager files are stored. This name is based on the queue manager name, but can be transformed if there is a duplicate name, or if the queue manager name is not a valid file name.

`Library=` ***`name`***
The name of the library where IBM i objects pertinent to this queue manager, for example, journals and journal receivers, are stored. This name is based on the queue manager name, but can be transformed if there is a duplicate name, or if the queue manager name is not a valid library name.

Changing queue manager configuration information on IBM i

IBM i

Use this information to understand the queue manager configuration stanzas.

There are two stanzas associated with API exits, ApiExitCommon and ApiExitTemplate. For details on using these stanzas, see Configuring API exits.

The following groups of attributes can occur in a `qm.ini` file for a specific queue manager, or used to override values set in `mqs.ini`.

See the following topics for changing configuration information for specific options:
- "The Log stanza on IBM i" on page 920
- "The Channels stanza on IBM i" on page 920
- "The queue manager error log stanza on IBM i" on page 922
- "The TCP stanza on IBM i" on page 923
- "PreConnect stanza of the client configuration file" on page 890

The Log stanza on IBM i: IBM i

Parameters for configuring the log file.

The `Log` stanza specifies the log attributes for a particular queue manager. By default, these attributes are inherited from the settings that are specified in the `LogDefaults` stanza in the `mqs.ini` file when the queue manager is created.

Change attributes of this stanza only if you want to configure a queue manager differently from others.

The values that are specified on the attributes in the `qm.ini` file are read when the queue manager is started. The file is created when the queue manager is created.

`LogBufferSize`
: The journal buffer size, in bytes. Enter a number in the range 32 000 - 15 761 440. The default is 32 000.

`LogPath=` *`library_name`*
: The name of the library that is used to store journals and journal receivers for this queue manager.

`LogReceiverSize`
: The journal receiver size, in kilobytes. The default is 100 000.

The Channels stanza on IBM i: IBM i

The `Channels` stanza contains information about the channels.

`MaxChannels= 100|`*`number`*
: The maximum number of *current* channels allowed. For z/OS, the value must be 1 - 9999, with a default value of 200. For all other platforms, the value must be 1 - 65,535, with a default value of 100.

`MaxActiveChannels=` *`MaxChannels_value`*
: The maximum number of channels allowed to be *active* at any time. The default is the value specified on the `MaxChannels` attribute.

`MaxInitiators= 3|`*`number`*
: The maximum number of initiators. The default and maximum value is 3.

`MQIBINDTYPE=FASTPATH|STANDARD`
: The binding for applications.

  `FASTPATH`
  : Channels connect using MQCONNX FASTPATH. That is, there is no agent process.

  `STANDARD`
  : Channels connect using STANDARD.

`ThreadedListener= NO|YES`
: Whether to start RUNMQLSR (`YES`) or AMQCLMAA (`NO`) as a listener.

  If you specify ThreadedListener=YES, all channels run as threads of a single job. This limits the number of connections to the resources available to a single job.

  If you specify ThreadedListener=NO, the non-threaded listener (AMQCLMAA) starts a new responder job (AMQCRSTA) for each inbound TCP/IP channel. The disadvantage of this technique is that it is not as fast to start a new AMQCRSTA job as it is to start a thread within a RUNMQLSR job, therefore connection times for a non-threaded listener are slower than those for a threaded listener.

`AdoptNewMCA= NO |SVR|SNDR|RCVR|CLUSRCVR|ALL|FASTPATH`
: If IBM MQ receives a request to start a channel, but finds that an amqcrsta process exists for the

same channel, the existing process must be stopped before the new one can start. The `AdoptNewMCA` attribute allows you to control the ending of an existing process and the startup of a new one for a specified channel type.

If you specify the `AdoptNewMCA` attribute for a given channel type, but the new channel fails to start because the channel is already running:

1. The new channel tries to end the previous one.
2. If the previous channel server does not end by the time the AdoptNewMCATimeout wait interval expires, the process (or the thread) for the previous channel server is ended.
3. If the previous channel server has not ended after step 2, and after the AdoptNewMCATimeout wait interval expires for a second time, IBM MQ ends the channel with a `CHANNEL IN USE` error.

You specify one or more values, separated by commas or blanks, from the following list:

NO The `AdoptNewMCA` feature is not required. This is the default.

SVR
Adopt server channels

SNDR
Adopt sender channels

RCVR
Adopt receiver channels

CLUSRCVR
Adopt cluster receiver channels

ALL
Adopt all channel types, except for FASTPATH channels

FASTPATH
Adopt the channel if it is a FASTPATH channel. This happens only if the appropriate channel type is also specified, for example, AdoptNewMCA=RCVR,SVR,FASTPATH

Attention!: The AdoptNewMCA attribute can behave in an unpredictable fashion with FASTPATH channels because of the internal design of the queue manager. Exercise great caution when enabling the `AdoptNewMCA` attribute for FASTPATH channels.

AdoptNewMCATimeout= 60|1-3600
The amount of time, in seconds, that the new process waits for the old process to end. Specify a value, in seconds, in the range 1 - 3600. The default value is 60.

AdoptNewMCACheck=QM|ADDRESS|NAME|ALL
The `AdoptNewMCACheck` attribute allows you to specify the type checking required when enabling the `AdoptNewMCA` attribute. It is important for you to perform all three of the following checks, if possible, to protect your channels from being shut down, inadvertently or maliciously. At the very least check that the channel names match.

Specify one or more values, separated by commas or blanks, from the following:

QM The listener process checks that the queue manager names match.

ADDRESS
The listener process checks the communications address, for example, the TCP/IP address.

NAME
The listener process checks that the channel names match.

ALL
The listener process checks for matching queue manager names, the communications address, and for matching channel names.

The default is `AdoptNewMCACheck=NAME,ADDRESS,QM`.

Related concepts:

"Channel states" on page 993
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.

The queue manager error log stanza on IBM i: IBM i

Use the `QMErrorLog` stanza in the `qm.ini` file to tailor the operation and contents of queue manager error logs.

V 9.0.4 **ErrorLogSize=** *maxsize*

Specifies the size of the queue manager error log which it is copied to the backup. *maxsize* must be in the range 32768 through 2147483648 bytes. If **ErrorLogSize** is not specified, the default value of 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.

Important: From IBM MQ Version 9.0.4, the default size of the **ErrorLogSize** attribute has increased. This is a change from IBM MQ Version 9.0.3.

ExcludeMessage= *msgIds*

Specifies messages that are not to be written to the queue manager error log. *msqIds* contain a comma-separated list of message IDs from the following:

7163 - Job started message (IBM i only)

7234 - Number of messages loaded

8245

9001 - Channel program ended normally

9002 - Channel program started

9202 - Remote host not available

9208 - Error on receive from host

9209 - Connection closed

9228 - Cannot start channel responder

9508 - Cannot connect to queue manager

9524 - Remote queue manager unavailable

9528 - User requested closure of channel

9558 - Remote Channel is not available

9776 - Channel was blocked by user id

9777 - Channel was blocked by NOACCESS map

9782 - Connection was blocked by address

9999 - Channel program ended abnormally

SuppressMessage= *msgIds*

Specifies messages that are written to the queue manager error log once only in a specified time interval. The time interval is specified by **SuppressInterval**. *msqIds* contain a comma-separated list of message IDs from the following:

7163 - Job started message (IBM i only)

7234 - Number of messages loaded

8245

9001 - Channel program ended normally

9002 - Channel program started
9202 - Remote host not available
9208 - Error on receive from host
9209 - Connection closed
9228 - Cannot start channel responder
9508 - Cannot connect to queue manager
9524 - Remote queue manager unavailable
9528 - User requested closure of channel
9558 - Remote Channel is not available
9776 - Channel was blocked by user id
9777 - Channel was blocked by NOACCESS map
9782 - Connection was blocked by address
9999 - Channel program ended abnormally

If the same message ID is specified in both **SuppressMessage** and **ExcludeMessage**, the message is excluded.

SuppressInterval= ***length***
Specifies the time interval, in seconds, in which messages specified in **SuppressMessage** 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.

The TCP stanza on IBM i: IBM i

Use these queue manager properties pages, or stanzas in the `qm.ini` file, to specify network protocol configuration parameters. They override the default attributes for channels.

Note: Only attributes representing changes to the default values need to be specified.

TCP
The following attributes can be specified:

Port= 1414|*port_number*
The default port number, in decimal notation, for TCP/IP sessions. The default port number from IBM MQ Version 8.0 is 1414.

KeepAlive= NO|YES
Switch the KeepAlive function on or off. KeepAlive=YES causes TCP/IP to check periodically that the other end of the connection is still available. If it is not, the channel is closed.

ListenerBacklog=number
When receiving on TCP/IP, a maximum number of outstanding connection requests is set. This can be considered to be a *backlog* of requests waiting on the TCP/IP port for the listener to accept the request. The default listener backlog value for IBM i is 255; the maximum is 512. If the backlog reaches the value of 512, the TCP/IP connection is rejected and the channel cannot start.

For MCA channels, this results in the channel going into a RETRY state and retrying the connection at a later time.

For client connections, the client receives an MQRC_Q_MGR_NOT_AVAILABLE reason code from MQCONN and should retry the connection at a later time.

The `ListenerBacklog` attribute allows you to override the default number of outstanding requests for the TCP/IP listener.

Connect_Timeout=*number*|0
The number of seconds before an attempt to connect the socket times out. The default value of zero specifies that there is no connect timeout.

The following group of properties can be used to control the size of buffers used by TCP/IP. The values are passed directly to the TCP/IP layer of the operating system. Great care should be taken when using these properties. If the values are set incorrectly it can adversely affect the TCP/IP performance. For further information about how this affects performance refer to the TCP/IP documentation for your environment. A value of zero indicates that the operating system will manage the buffer sizes, as opposed to the buffer sizes being fixed by IBM MQ.

SndBuffSize=*number*|0
The size in bytes of the TCP/IP send buffer used by the sending end of channels. This stanza value can be overridden by a stanza more specific to the channel type, for example RcvSndBuffSize. 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.

RcvBuffSize=*number*|0
The size in bytes of the TCP/IP receive buffer used by the receiving end of channels. This stanza value can be overridden by a stanza more specific to the channel type, for example RcvRcvBuffSize. If the value is set as zero, the operating system defaults are used. If no value is set, then the IBM MQ default, 32768, is used.

RcvSndBuffSize=*number*|0
The size in bytes of the TCP/IP send buffer used by the sender end of a receiver channel. If the value is set as zero, the operating system defaults are used. If no value is set, then the IBM MQ default, 32768, is used.

RcvRcvBuffSize=*number*|0
The size in bytes of the TCP/IP receive buffer used by the receiving end of a receiver channel. If the value is set as zero, the operating system defaults are used. If no value is set, then the IBM MQ default, 32768, is used.

SvrSndBuffSize=*number*|0
The size in bytes of the TCP/IP send buffer used by the server end of a client-connection server-connection channel. If the value is set as zero, the operating system defaults are used. If no value is set, then the IBM MQ default, 32768, is used.

SvrRcvBuffSize=*number*|0
The size in bytes of the TCP/IP receive buffer used by the server end of a client-connection server-connection channel. If the value is set as zero, the operating system defaults are used. If no value is set, then the IBM MQ default, 32768, is used.

The tuning parameters stanza on IBM i: IBM i

Use the `TuningParameters` stanza in the `qm.ini` file to specify options for tuning the queue manager.

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.

Related information:
Implicit syncpoint

Example mqs.ini and qm.ini files for IBM i

IBM i

Figure 82 shows an example of an mqs.ini file for IBM i.

```
#*******************************************************************#
#* Module Name: mqs.ini                                            *#
#* Type       : IBM MQ Configuration File                          *#
#* Function   : Define IBM MQ resources for the node               *#
#*                                                                 *#
#*******************************************************************#
#* Notes      :                                                    *#
#* 1) This is an example IBM MQ configuration file                 *#
#*                                                                 *#
#*******************************************************************#
AllQueueManagers:
#*******************************************************************#
#* The path to the qmgrs directory, within which queue manager data *#
#* is stored                                                       *#
#*******************************************************************#
DefaultPrefix=/QIBM/UserData/mqm

QueueManager:
Name=saturn.queue.manager
Prefix=/QIBM/UserData/mqm
Library=QMSATURN.Q
Directory=saturn!queue!manager

QueueManager:
Name=pluto.queue.manager
Prefix=/QIBM/UserData/mqm
Library=QMPLUTO.QU
Directory=pluto!queue!manager

DefaultQueueManager:
Name=saturn.queue.manager
```

Figure 82. Example of an IBM MQ configuration file

Figure 83 on page 926 shows how groups of attributes might be arranged in a queue manager configuration file for IBM i.

V 9.0.5 From IBM MQ Version 9.0.5, this is an example queue manager configuration file:

```
#*******************************************************************#
#* Module Name: qm.ini                                             *#
#* Type       : IBM MQ queue manager configuration file            *#
# Function    : Define the configuration of a single queue manager *#
#*                                                                 *#
#*******************************************************************#
#* Notes      :                                                    *#
#* 1) This file defines the configuration of the queue manager     *#
#*                                                                 *#
#*******************************************************************#
Log:
LogPath=QMSATURN.Q
LogReceiverSize=65536

CHANNELS:
MaxChannels = 20       ; Maximum number of channels allowed.
                       ; Default is 100.
MaxActiveChannels = 10 ; Maximum number of channels allowed to be
                       ; active at any time. The default is the
                       ; value of MaxChannels.

TCP:                   ; TCP/IP entries.
KeepAlive = Yes        ; Switch KeepAlive on.
SvrSndBuffSize=20000   ; Size in bytes of the TCP/IP send buffer for each
                       ; channel instance. Default is 32768.
SvrRcvBuffSize=20000   ; Size in bytes of the TCP/IP receive buffer for each
                       ; channel instance. Default is 32768.
Connect_Timeout=10000  ; Number of seconds before an attempt to connect the
                       ; channel instance times out. Default is zero (no timeout).

QMErrorLog:
ErrorLogSize = 262144
ExcludeMessage = 7234
SuppressMessage = 9001,9002,9202
SuppressInterval = 30

TuningParameters:
   ImplSyncOpenOutput=2
```

For IBM MQ Version 9.0.4 and earlier, and LTS, this is an example configuration file:

```
#*******************************************************************#
#* Module Name: qm.ini                                             *#
#* Type       : IBM MQ queue manager configuration file            *#
# Function    : Define the configuration of a single queue manager *#
#*                                                                 *#
#*******************************************************************#
#* Notes      :                                                    *#
#* 1) This file defines the configuration of the queue manager     *#
#*                                                                 *#
#*******************************************************************#
Log:
LogPath=QMSATURN.Q
LogReceiverSize=65536

CHANNELS:
MaxChannels = 20       ; Maximum number of channels allowed.
                       ; Default is 100.
MaxActiveChannels = 10 ; Maximum number of channels allowed to be
                       ; active at any time. The default is the
                       ; value of MaxChannels.

TCP:                   ; TCP/IP entries.
KeepAlive = Yes        ; Switch KeepAlive on.
SvrSndBuffSize=20000   ; Size in bytes of the TCP/IP send buffer for each
                       ; channel instance. Default is 32768.
SvrRcvBuffSize=20000   ; Size in bytes of the TCP/IP receive buffer for each
                       ; channel instance. Default is 32768.
Connect_Timeout=10000  ; Number of seconds before an attempt to connect the
                       ; channel instance times out. Default is zero (no timeout).
```

Note:

1. IBM MQ on the node uses the default locations for queue managers and the journals.
2. The queue manager saturn.queue.manager is the default queue manager for the node. The directory for files associated with this queue manager has been automatically transformed into a valid file name for the file system.
3. Because the IBM MQ configuration file is used to locate the data associated with queue managers, a nonexistent or incorrect configuration file can cause some or all IBM MQ commands to fail. Also, applications cannot connect to a queue manager that is not defined in the IBM MQ configuration file.

Attributes for changing IBM MQ configuration information

On IBM MQ for Windows systems and on IBM MQ for Linux (x86 and x86-64 platforms) systems, modify configuration information using the IBM MQ Explorer. On other systems, modify the information by editing the mqs.ini configuration file.

See the following subtopics for attributes for specific components:

Related concepts:

"Configuring IBM MQ" on page 833
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.

Related tasks:

"Changing IBM MQ and queue manager configuration information" on page 906
You can change the behavior of IBM MQ or an individual queue manager to suit the needs of your installation.

"Changing queue manager configuration information" on page 934
The attributes that you can use to modify the configuration of an individual queue manager override any settings for IBM MQ.

Related information:

Planning

Administering IBM MQ

All queue managers

Use the `General` and `Extended` IBM MQ properties page from the IBM MQ Explorer, or the `AllQueueManagers` stanza in the mqs.ini file to specify the following information about all queue managers.

`DefaultPrefix=` ***`directory_name`***

This attribute specifies the path to the qmgrs directory, within which the queue manager data is kept.

If you change the default prefix for the queue manager, replicate the directory structure that was created at installation time.

In particular, you must create the qmgrs structure. Stop IBM MQ before changing the default prefix, and restart IBM MQ only after you have moved the structures to the new location and changed the default prefix.

Note: Do not delete the `/var/mqm/errors` directory on UNIX and Linux systems, or the `\errors` directory on Windows systems.

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

Because of operating system restrictions, keep the supplied path sufficiently short so that the sum of the path length and any queue manager name is a maximum of 70 characters long.

ConvEBCDICNewline=NL_TO_LF|TABLE|ISO

EBCDIC code pages contain a newline (NL) character that is not supported by ASCII code pages (although some ISO variants of ASCII contain an equivalent).

Use the ConvEBCDICNewline attribute to specify how IBM MQ is to convert the EBCDIC NL character into ASCII format.

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 109.

Table 109. List of possible ISO CCSIDs

CCSID	Code Set
819	ISO8859-1
912	ISO8859-2
915	ISO8859-5
1089	ISO8859-6
813	ISO8859-7
916	ISO8859-8
920	ISO8859-9
1051	roman8

If the ASCII CCSID is not an ISO subset, ConvEBCDICNewline defaults to NL_TO_LF.

Default queue manager

Use the `General` IBM MQ properties page from the IBM MQ Explorer, or the `DefaultQueueManager` stanza in the mqs.ini file to specify the default queue manager.

`Name=` ***`default_queue_manager`***

The default queue manager processes any commands for which a queue manager name is not explicitly specified. The `DefaultQueueManager` attribute is automatically updated if you create a new default queue manager. If you inadvertently create a new default queue manager and then want to revert to the original, alter the `DefaultQueueManager` attribute manually.

Exit properties

Use the `Extended` IBM MQ properties page from the IBM MQ Explorer, or the `ExitProperties` stanza in the mqs.ini file to specify configuration options used by queue manager exit programs.

`CLWLMode= SAFE|FAST`

The cluster workload (CLWL) exit allows you to specify which cluster queue in the cluster to open in response to an MQI call (for example, `MQOPEN`, `MQPUT`). The CLWL exit runs either in FAST mode or SAFE mode depending on the value you specify on the CLWLMode attribute. If you omit the CLWLMode attribute, the cluster workload exit runs in SAFE mode.

`SAFE`

Run the CLWL exit in a separate process from the queue manager. This is the default.

If a problem arises with the user-written CLWL exit when running in SAFE mode, the following happens:

- The CLWL server process (amqzlwa0) fails.
- The queue manager restarts the CLWL server process.
- The error is reported to you in the error log. If an MQI call is in progress, you receive notification in the form of a return code.

The integrity of the queue manager is preserved.

Note: Running the CLWL exit in a separate process can affect performance.

`FAST`

Run the cluster exit inline in the queue manager process.

Specifying this option improves performance by avoiding the process switching costs associated with running in SAFE mode, but does so at the expense of queue manager integrity. You should only run the CLWL exit in FAST mode if you are convinced that there are **no** problems with your CLWL exit, and you are particularly concerned about performance.

If a problem arises when the CLWL exit is running in FAST mode, the queue manager will fail and you run the risk of the integrity of the queue manager being compromised.

Log defaults for IBM MQ

Use the `Default log settings` IBM MQ properties page from the IBM MQ Explorer, or the `LogDefaults` stanza in the mqs.ini file to specify information about log defaults for all queue managers.

If the stanza does not exist then the MQ defaults will be used. The log attributes are used as default values when you create a queue manager, but can be overridden if you specify the log attributes on the `crtmqm` command. See **`crtmqm`** for details of this command.

Once a queue manager has been created, the log attributes for that queue manager are taken from the settings described in "Queue manager logs" on page 938.

The default prefix (specified in "All queue managers" on page 927) and log path specified for the particular queue manager (specified in "Queue manager logs" on page 938) 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 1331.

Note: The limits given in the following parameter list are limits set by IBM MQ. Operating system limits might reduce the maximum possible log size.

LogPrimaryFiles= 3|*2-254* (Windows)|*2-510* (UNIX and Linux systems)
The log files allocated when the queue manager is created.

The minimum number of primary log files you can have is 2 and the maximum is 254 on Windows, or 510 on UNIX and Linux. The default is 3.

The total number of primary and secondary log files must not exceed 255 on Windows, or 511 on UNIX and Linux, and must not be less than 3.

The value is examined when the queue manager is created or started. You can change it after the queue manager has been created. However, a change in the value is not effective until the queue manager is restarted, and the effect might not be immediate.

LogSecondaryFiles= 2|*1-253* (Windows)|*1-509* (UNIX and Linux)
The log files allocated when the primary files are exhausted.

The minimum number of secondary log files is 1 and the maximum is 253 on Windows, or 509 on UNIX and Linux. The default number is 2.

The total number of primary and secondary log files must not exceed 255 on Windows, or 511 on UNIX and Linux, and must not be less than 3.

The value is examined when the queue manager is started. You can change this value, but changes do not become effective until the queue manager is restarted, and even then the effect might not be immediate.

LogFilePages= *number*
The log data is held in a series of files called log files. The log file size is specified in units of 4 KB pages.

The default number of log file pages is 4096, giving a log file size of 16 MB.

On UNIX and Linux, the minimum number of log file pages is 64, and on Windows, the minimum number of log file pages is 32; in both cases the maximum number is 65 535.

Note: The size of the log files specified during queue manager creation cannot be changed for a queue manager.

LogType= CIRCULAR|LINEAR
The type of log to be used. The default is CIRCULAR.

CIRCULAR
Start restart recovery using the log to roll back transactions that were in progress when the system stopped.

See "Types of logging" on page 1325 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 1325 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.

V 9.0.4 From IBM MQ Version 9.0.4, you can change the logging method after a queue manager has been created. See migmqlog for more information.

LogBufferPages= 0|*0-4096*
The amount of memory allocated to buffer records for writing, specifying the size of the buffers in units of 4 KB pages.

The minimum number of buffer pages is 18 and the maximum is 4096. Larger buffers lead to higher throughput, especially for larger messages.

If you specify 0 (the default), the queue manager selects the size. In IBM WebSphere MQ Version 7.1 this is 512 (2048 KB).

If you specify a number in the range 1 through 17, the queue manager defaults to 18 (72 KB). If you specify a number in the range 18 and through 4096, the queue manager uses the number specified to set the memory allocated.

LogDefaultPath= *directory_name*
The directory in which the log files for a queue manager reside. The directory resides on a local device to which the queue manager can write and, preferably, on a different drive from the message queues. Specifying a different drive gives added protection in case of system failure.

The default is:

- *`DefaultPrefix`*`\log` for IBM MQ for Windows where *`DefaultPrefix`* is the value specified on the `DefaultPrefix` attribute on the `All Queue Managers` IBM MQ properties page. This value is set at installation time.
- `/var/mqm/log` for IBM MQ for UNIX and Linux systems

Alternatively, you can specify the name of a directory on the `crtmqm` command using the -ld flag. When a queue manager is created, a directory is also created under the queue manager directory, and this is used to hold the log files. The name of this directory is based on the queue manager name. This ensures that the log file path is unique, and also that it conforms to any limitations on directory name lengths.

If you do not specify -ld on the `crtmqm` command, the value of the `LogDefaultPath` attribute in the mqs.ini file is used.

The queue manager name is appended to the directory name to ensure that multiple queue managers use different log directories.

When the queue manager is created, a `LogPath` value is created in the log attributes in the configuration information, giving the complete directory name for the queue manager's log. This value is used to locate the log when the queue manager is started or deleted.

LogWriteIntegrity=SingleWrite|DoubleWrite|TripleWrite
The method the logger uses to reliably write log records.

TripleWrite
This is the default method.

Note, that you can select **DoubleWrite**, but if you do so, the system interprets this as **TripleWrite**.

SingleWrite
You should use **SingleWrite**, only if the file-system and device hosting the IBM MQ recovery log explicitly guarantees the atomicity of 4KB writes.

That is, when a write of a 4KB page fails for any reason, the only two possible states are either the before image, or the after image. No intermediate state should be possible.

Note: If there is sufficient concurrency in your persistent workload, there is minimal potential benefit in setting anything other than the default value, **TripleWrite**.

Advanced Configuration and Power Interface (ACPI)

Use the `ACPI` IBM MQ properties page from the IBM MQ Explorer, to specify how IBM MQ is to behave when the system receives a suspend request.

Windows supports the Advanced Configuration and Power Interface (ACPI) standard. This enables Windows users with ACPI enabled hardware to stop and restart channels when the system enters and resumes from suspend mode.

Note that the settings specified in the `ACPI` IBM MQ properties page are applied only when the Alert Monitor is running. The Alert Monitor icon is present on the taskbar if the Alert Monitor is running.

DoDialog= Y | N
Displays the dialog at the time of a suspend request.

DenySuspend=Y | N
Denies the suspend request. This is used if DoDialog=N, or if DoDialog=Y and a dialog cannot be displayed, for example, because your notebook lid is closed.

CheckChannelsRunning=Y | N
Checks whether any channels are running. The outcome can determine the outcome of the other settings.

The following table outlines the effect of each combination of these parameters:

DoDialog	DenySuspend	CheckChannels Running	Action
N	N	N	Accept the suspend request.
N	N	Y	Accept the suspend request.
N	Y	N	Deny the suspend request.
N	Y	Y	If any channels are running deny the suspend request; if not accept the request.
Y	N	N	Display the dialog (see Note ; accept the suspend request). This is the default.
Y	N	Y	If no channels are running accept the suspend request; if they are display the dialog (see Note ; accept the request).
Y	Y	N	Display the dialog (Note ; deny the suspend request).
Y	Y	Y	If no channels are running accept the suspend request; if they are display the dialog (Note ; deny the request).

Note: In cases where the action is to display the dialog, if the dialog cannot be displayed (for example because your notebook lid is closed), the DenySuspend option is used to determine whether the suspend request is accepted or denied.

API exits

Use the IBM MQ Explorer or the `amqmdain` command to change the entries for API exits.

Use the `Exits` IBM MQ properties page from the IBM MQ Explorer, or the `ApiExitTemplate` and `ApiExitCommon` stanza in the mqs.ini file to identify API exit routines for all queue managers. On Windows systems, you can also use the `amqmdain` command to change the entries for API exits. (To identify API exit routines for individual queue managers, you use the `ApiExitLocal` stanza, as described in "Queue manager configuration files, qm.ini" on page 911.)

For a complete description of the attributes for these stanzas, see Configuring API exits.

Queue managers

There is one `QueueManager` stanza for every queue manager. Use the stanza to specify the location of the queue manager directory.

On Windows, UNIX and Linux systems, there is one `QueueManager` stanza for every queue manager. These attributes specify the queue manager name, and the name of the directory containing the files associated with that queue manager. The name of the directory is based on the queue manager name, but is transformed if the queue manager name is not a valid file name. See, Understanding IBM MQ file names for more information about name transformation.

`Name=` ***`queue_manager_name`***
: The name of the queue manager.

`Prefix=` ***`prefix`***
: Where the queue manager files are stored. By default, this value is the same as the value specified on the DefaultPrefix attribute of the All Queue Managers information.

`Directory=` ***`name`***
: The name of the subdirectory under the *`prefix`*`\QMGRS` directory where the queue manager files are stored. This name is based on the queue manager name, but can be transformed if there is a duplicate name or if the queue manager name is not a valid file name.

`DataPath=` ***`path`***
: An explicit data path provided when the queue manager was created, this overrides Prefix and Directory as the path to the queue manager data.

`InstallationName=` ***`name`***
: The name of the IBM MQ installation associated with this queue manager. Commands from this installation must be used when interacting with this queue manager. If no InstallationName value is present, the queue manager is associated with an installation of the product earlier than Version 7.1.

Related tasks:

"Associating a queue manager with an installation" on page 1196
When you create a queue manager, it is automatically associated with the installation that issued the **crtmqm** command. On UNIX, Linux, and Windows, you can change the installation associated with a queue manager using the **setmqm** command.

Changing queue manager configuration information

The attributes that you can use to modify the configuration of an individual queue manager override any settings for IBM MQ.

About this task

On UNIX and Linux systems, you modify queue manager configuration information by editing the `qm.ini` configuration file. When you are defining a stanza in `qm.ini`, you do not need to start each item on a new line. You can use either a semicolon (;) or a hash character (#) to indicate a comment.

On Windows and Linux x86-64 systems, you can modify some configuration information by using the IBM MQ Explorer. However, because there are significant implications to changing installable services and their components, the installable services are read-only in the IBM MQ Explorer. You must therefore make any changes to installable services by using **regedit** on Windows, and by editing the `qm.ini` file on UNIX and Linux.

Procedure

For more details on changing queue manager configuration information, see the following subtopics:.

Related concepts:

"Configuring IBM MQ" on page 833
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.

Related tasks:

"Changing IBM MQ and queue manager configuration information" on page 906
You can change the behavior of IBM MQ or an individual queue manager to suit the needs of your installation.

Related reference:

"Attributes for changing IBM MQ configuration information" on page 927
On IBM MQ for Windows systems and on IBM MQ for Linux (x86 and x86-64 platforms) systems, modify configuration information using the IBM MQ Explorer. On other systems, modify the information by editing the mqs.ini configuration file.

Related information:

Planning

Administering IBM MQ

Access Mode

Windows

`Access Mode` applies to Windows servers only. The AccessMode stanza is set by the `-a [r]` option on the **crtmqm** command. Do not change the AccessMode stanza after the queue manager has been created.

Use the access group (`-a [r]`) option of the **crtmqm** command to specify a Windows security group, members of which will be granted full access to all queue manager data files. The group can either be a local or global group, depending upon the syntax used. Valid syntax for the group name is as follows:

LocalGroup

Domain name\GlobalGroup name

GlobalGroup name @ Domain name

You must define the additional access group before running the crtmqm command with the `-a [r]` option.

If you specify the group using `-ar` instead of `-a`, the local mqm group is not granted access to the queue manager data files. Use this option, if the file system hosting the queue manager data files does not support access control entries for locally defined groups.

The group is typically a global security group, which is used to provide multi-instance queue managers with access to a shared queue manager data and logs folder. Use the additional security access group to set read and write permissions on the folder or to share containing queue manager data and log files.

The additional security access group is an alternative to using the local group named `mqm` to set permissions on the folder containing queue manager data and logs. Unlike the local group `mqm`, you can make the additional security access group a local or a global group. It must be a global group to set permissions on the shared folders that contain the data and log files used by multi-instance queue managers.

The Windows operating system checks the access permissions to read and write queue manager data and log files. It checks the permissions of the user ID that is running queue manager processes. The user ID that is checked depends on whether you started the queue manager as a service or you started it interactively. If you started the queue manager as a service, the user ID checked by the Windows system is the user ID you configured with the Prepare IBM MQ wizard. If you started the queue manager interactively, the user ID checked by the Windows system is the user ID that ran the **`strmqm`** command.

The user ID must be a member of the local `mqm` group to start the queue manager. If the user ID is a member of the additional security access group, the queue manager can read and write files that are given permissions by using the group.

Restriction: You can specify an additional security access group only on Windows operating system. If you specify an additional security access group on other operating systems, the **`crtmqm`** command returns an error.

Related concepts:

"Secure unshared queue manager data and log directories and files on Windows" on page 1269
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.

"Securing shared queue manager data and log directories and files on Windows" on page 1266
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.

Related tasks:

"Creating a multi-instance queue manager on domain workstations or servers on Windows" on page 1243
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.

Related information:

crtmqm

Configuring installable services

ULW

You change installable services on Windows by using **regedit**, and on UNIX and Linux by using the `Service` stanza in the `qm.ini` file.

Note: There are significant implications to changing installable services and their components. For this reason, the installable services are read-only in the IBM MQ Explorer.

To change installable services on Windows systems, use **regedit**, or on UNIX and Linux systems, use the `Service` stanza in the `qm.ini` file. For each component within a service, you must also specify the name and path of the module containing the code for that component. On UNIX and Linux systems, use the `ServiceComponent` stanza for this.

Name= AuthorizationService|NameService
The name of the required service.

AuthorizationService
For IBM MQ, the Authorization Service component is known as the object authority manager, or OAM. The `AuthorizationService` stanza and its associated `ServiceComponent` stanza are added automatically when the queue manager is created. Add other `ServiceComponent` stanzas manually.

NameService
No name service is provided by default. If you require a name service, you must add the `NameService` stanza manually.

EntryPoints= ***number-of-entries***
The number of entry points defined for the service.

This includes the initialization and termination entry points.

Windows **SecurityPolicy= 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.

See Windows security identifiers (SIDs) for more information.

See also Configuring authorization service stanzas: Windows systems.

Linux UNIX **SecurityPolicy=user|group|default**
On UNIX and Linux systems the value specifies whether the queue manager uses user-based or group-based authorization. Values are not case sensitive.

If you do not include this attribute, `default` is used, which uses group-based authorization. Restart the queue manager for changes to become effective. See also Configuring authorization service stanzas: UNIX and Linux systems.

SharedBindingsUserId= ***user-type***
The SharedBindingsUserId attribute applies only if the service specified is the default authorization service, that is, the OAM. The SharedBindingsUserId attribute is used with relation to shared bindings only. This value allows you to specify whether the *UserIdentifier* field in the *IdentityContext* structure, from the MQZ_AUTHENTICATE_USER function, is the effective user Id or the real user Id.

For information on the MQZ_AUTHENTICATE_USER function, see MQZ_AUTHENTICATE_USER - Authenticate user.

The possible values are:

Default
The value of the *UserIdentifier* field is set as the real user Id.

Real
The value of the *UserIdentifier* field is set as the real user Id.

Effective
The value of the *UserIdentifier* field is set as the effective user Id.

FastpathBindingsUserId= ***user-type***
The FastpathBindingsUserId attribute applies only if the service specified is the default authorization service, that is, the OAM. The FastpathBindingsUserId attribute is used with relation to fastpath bindings only. This value allows you to specify whether the *UserIdentifier* field in the *IdentityContext* structure, from the MQZ_AUTHENTICATE_USER function, is the effective user Id or the real user Id.

For information on the MQZ_AUTHENTICATE_USER function, see MQZ_AUTHENTICATE_USER - Authenticate user.

The possible values are:

Default
The value of the *UserIdentifier* field is set as the real user ID.

Real
The value of the *UserIdentifier* field is set as the real user ID.

Effective
The value of the *UserIdentifier* field is set as the effective user ID.

IsolatedBindingsUserId= ***user-type***
The **IsolatedBindingsUserId** attribute applies only if the service specified is the default authorization service, that is, the OAM. The **IsolatedBindingsUserId** attribute is used with relation to isolated bindings only. This value allows you to specify whether the *UserIdentifier* field in the *IdentityContext* structure, from the MQZ_AUTHENTICATE_USER function, is the effective user Id or the real user Id.

For information on the MQZ_AUTHENTICATE_USER function, see MQZ_AUTHENTICATE_USER - Authenticate user.

The possible values are:

Default
The value of the *UserIdentifier* field is set as the effective user Id.

Real
The value of the *UserIdentifier* field is set as the real user Id.

Effective
The value of the *UserIdentifier* field is set as the effective user Id.

For more information about installable services and components, see Installable services and components for UNIX, Linux, and Windows.

For more information about security services in general, see Setting up security on UNIX and Linux systems.

Related information:
Installable services reference information

Service components: ULW

You must specify service component information when you add a new installable service. On Windows systems use **`regedit`**, and on UNIX and Linux systems use the **`ServiceComponent`** stanza in the qm.ini file. The authorization service stanza is present by default, and the associated component, the OAM, is active.

Specify the service components as follows:

`Service=` ***`service_name`***
The name of the required service. This must match the value specified on the `Name` attribute of the Service configuration information.

`Name=` ***`component_name`***
The descriptive name of the service component. This must be unique and contain only characters that are valid for the names of IBM MQ objects (for example, queue names). This name occurs in operator messages generated by the service. We recommend that this name begins with a company trademark or similar distinguishing string.

`Module=` ***`module_name`***
The name of the module to contain the code for this component. This must be a full path name.

`ComponentDataSize=` ***`size`***
The size, in bytes, of the component data area passed to the component on each call. Specify zero if no component data is required.

For more information about installable services and components, see Installable services and components for UNIX, Linux and Windows.

Queue manager logs

ULW

Use the `Log` queue manager properties page from the IBM MQ Explorer, or the `Log` stanza in the qm.ini file, to specify information about logging on a queue manager.

By default, these settings are inherited from the settings specified for the default log settings for the queue manager (described in “Log defaults for IBM MQ” on page 930). Change these settings only if you want to configure this queue manager in a different way.

For information about calculating log sizes, see “Calculating the size of the log” on page 1331.

Note: The limits given in the following parameter list are set by IBM MQ. Operating system limits might reduce the maximum possible log size.

`LogPrimaryFiles= 3|`*`2-254`*` ( Windows )|`*`2-510`*` ( UNIX and Linux systems)`
The log files allocated when the queue manager is created.

The minimum number of primary log files you can have is 2 and the maximum is 254 on Windows, or 510 on UNIX and Linux systems. The default is 3.

The total number of primary and secondary log files must not exceed 255 on Windows, or 511 on UNIX and Linux systems, and must not be less than 3.

The value is examined when the queue manager is created or started. You can change it after the queue manager has been created. However, a change in the value is not effective until the queue manager is restarted, and the effect might not be immediate.

LogSecondaryFiles= 2|*1-253* (Windows)|*1-509* (UNIX and Linux systems)
The log files allocated when the primary files are exhausted.

The minimum number of secondary log files is 1 and the maximum is 253 on Windows, or 509 on UNIX and Linux systems. The default number is 2.

The total number of primary and secondary log files must not exceed 255 on Windows, or 511 on UNIX and Linux systems, and must not be less than 3.

The value is examined when the queue manager is started. You can change this value, but changes do not become effective until the queue manager is restarted, and even then the effect might not be immediate.

LogFilePages= *number*
The log data is held in a series of files called log files. The log file size is specified in units of 4 KB pages.

The default number of log file pages is 4096, giving a log file size of 16 MB.

On UNIX and Linux systems the minimum number of log file pages is 64, and on Windows the minimum number of log file pages is 32; in both cases the maximum number is 65 535.

Note: The size of the log files specified during queue manager creation cannot be changed for a queue manager.

LogType= CIRCULAR|LINEAR
The type of logging to be used by the queue manager. You cannot change the type of logging to be used once the queue manager has been created. Refer to the description of the LogType attribute in "Log defaults for IBM MQ" on page 930 for information about creating a queue manager with the type of logging you require.

CIRCULAR
Start restart recovery using the log to roll back transactions that were in progress when the system stopped.

See "Types of logging" on page 1325 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 1325 for a fuller explanation of linear logging.

LogBufferPages= 0|*0-4096*
The amount of memory allocated to buffer records for writing, specifying the size of the buffers in units of 4 KB pages.

The minimum number of buffer pages is 18 and the maximum is 4096. Larger buffers lead to higher throughput, especially for larger messages.

If you specify 0 (the default), the queue manager selects the size. In IBM WebSphere MQ Version 7.1 this is 512 (2048 KB).

If you specify a number in the range 1 through 17, the queue manager defaults to 18 (72 KB). If you specify a number in the range 18 through 4096, the queue manager uses the number specified to set the memory allocated.

The value is examined when the queue manager is started. The value can be increased or decreased within the limits stated. However, a change in the value is not effective until the next time the queue manager is started.

LogPath= *directory_name*
The directory in which the log files for a queue manager reside. This must exist on a local device to

which the queue manager can write and, preferably, on a different drive from the message queues. Specifying a different drive gives added protection in case of system failure.

The default is:

- `C:\ProgramData\IBM\MQ\log` in IBM MQ for Windows.
- `/var/mqm/log` in IBM MQ for UNIX and Linux systems.

You can specify the name of a directory on the `crtmqm` command using the -ld flag. When a queue manager is created, a directory is also created under the queue manager directory, and this is used to hold the log files. The name of this directory is based on the queue manager name. This ensures that the log file path is unique, and also that it conforms to any limitations on directory name lengths.

If you do not specify -ld on the `crtmqm` command, the value of the `LogDefaultPath` attribute is used.

In IBM MQ for UNIX and Linux systems, user ID mqm and group mqm must have full authorities to the log files. If you change the locations of these files, you must give these authorities yourself. This is not required if the log files are in the default locations supplied with the product.

`LogWriteIntegrity=SingleWrite|DoubleWrite|TripleWrite`
The method the logger uses to reliably write log records.

`TripleWrite`
This is the default method.

Note, that you can select **DoubleWrite**, but if you do so, the system interprets this as **TripleWrite**.

`SingleWrite`
You should use **SingleWrite**, only if the file-system and device hosting the IBM MQ recovery log explicitly guarantees the atomicity of 4KB writes.

That is, when a write of a 4KB page fails for any reason, the only two possible states are either the before image, or the after image. No intermediate state should be possible.

Note: If there is sufficient concurrency in your persistent workload, there is minimal potential benefit in setting anything other than the default value, **TripleWrite**.

V 9.0.2 **`LogManagement= Manual|Automatic | Archive`**
The method used to manage log extents, either manually or by the queue manager.

The attribute applies only when **`LogType`** is LINEAR.
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.

`Manual`
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.

Restricted mode

UNIX Linux

This option applies to UNIX and Linux systems only. The `RestrictedMode` stanza is set by the **`-g`** option on the **`crtmqm`** command. Do not change this stanza after the queue manager has been created. If you do not use the **`-g`** option, the stanza is not created in the `qm.ini` file.

There are some directories under which IBM MQ applications create files while they are connected to the queue manager within the queue manager data directory. In order for applications to create files in these directories, they are granted world write access:
- `/var/mqm/sockets/`*`QMgrName`*`/@ipcc/ssem/`*`hostname`*`/`
- `/var/mqm/sockets/`*`QMgrName`*`/@app/ssem/`*`hostname`*`/`
- `/var/mqm/sockets/`*`QMgrName`*`/zsocketapp/`*`hostname`*`/`

where *`QMGRNAME`* is the name of the queue manager, and *`hostname`* is the host name.

On some systems, it is unacceptable to grant all users write access to these directories. For example, those users who do not need access the queue manager. Restricted mode modifies the permissions of the directories that store queue manager data. The directories can then only be accessed by members of the specified application group. The permissions on the System V IPC shared memory used to communicate with the queue manager are also modified in the same way.

The application group is the name of the group with members that have permission to do the following things:
- Run MQI applications
- Update all IPCC resources
- Change the contents of some queue manager directories

To use restricted mode for a queue manager:
- The creator of the queue manager must be in the `mqm` group and in the application group.
- The `mqm` user ID must be in the application group.
- All users who want to administer the queue manager must be in the `mqm` group and in the application group.
- All users who want to run IBM MQ applications must be in the application group.

Any MQCONN or MQCONNX call issued by a user who is not in the application group 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 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.

ULW Further information about configuring the IBM MQ authorization service can be found in Setting up security on Windows, UNIX and Linux systems.

Only use IBM MQ restricted mode when the control provided by the authorization service does not provide sufficient isolation of queue manager resources.

XA resource managers

Use the `XA resource manager` queue manager properties page from the IBM MQ Explorer, or the `XAResourceManager` stanza in the qm.ini file, to specify the following information about the resource managers involved in global units of work coordinated by the queue manager.

Add XA resource manager configuration information manually for each instance of a resource manager participating in global units of work; no default values are supplied.

See Database coordination for more information about resource manager attributes.

`Name=` *`name`* `(mandatory)`
This attribute identifies the resource manager instance.

The `Name` value can be up to 31 characters in length. You can use the name of the resource manager as defined in its XA-switch structure. However, if you are using more than one instance of the same resource manager, you must construct a unique name for each instance. You can ensure uniqueness by including the name of the database in the `Name` string, for example.

IBM MQ uses the `Name` value in messages and in output from the `dspmqtrn` command.

Do not change the name of a resource manager instance, or delete its entry from the configuration information, once the associated queue manager has started and the resource manager name is in effect.

`SwitchFile=` *`name`* `(mandatory)`
The fully-qualified name of the load file containing the resource manager's XA switch structure.

If you are using a 64-bit queue manager with 32-bit applications, the `name` value should contain only the base name of the load file containing the resource manager's XA switch structure.

The 32-bit file will be loaded into the application from the path specified by `ExitsDefaultPath`.

The 64-bit file will be loaded into the queue manager from the path specified by `ExitsDefaultPath64`.

`XAOpenString=` *`string`* `(optional)`
The string of data to be passed to the resource manager's xa_open entry point. The contents of the string depend on the resource manager itself. For example, the string could identify the database that this instance of the resource manager is to access. For more information about defining this attribute, see:

- Adding resource manager configuration information for Db2
- Adding resource manager configuration information for Oracle
- Adding resource manager configuration information for Sybase
- Adding resource manager configuration information for Informix®

and consult your resource manager documentation for the appropriate string.

`XACloseString=` *`string`* `(optional)`
The string of data to be passed to the resource manager's xa_close entry point. The contents of the string depend on the resource manager itself. For more information about defining this attribute, see:

- Adding resource manager configuration information for Db2
- Adding resource manager configuration information for Oracle
- Adding resource manager configuration information for Sybase
- Adding resource manager configuration information for Informix

and consult your database documentation for the appropriate string.

`ThreadOfControl=THREAD|PROCESS`
Windows This attribute is mandatory for IBM MQ for Windows. The queue manager uses this value for serialization when it needs to call the resource manager from one of its own multithreaded processes.

THREAD
The resource manager is fully *thread aware*. In a multithreaded IBM MQ process, XA function calls can be made to the external resource manager from multiple threads at the same time.

PROCESS
The resource manager is not *thread safe*. In a multithreaded IBM MQ process, only one XA function call at a time can be made to the resource manager.

The `ThreadOfControl` entry does not apply to XA function calls issued by the queue manager in a multithreaded application process. In general, an application that has concurrent units of work on different threads requires this mode of operation to be supported by each of the resource managers.

Attributes of the channels stanza

These attributes determine the configuration of a channel.

This information is not applicable to IBM MQ for the z/OS platform.

Use the `Channels` queue manager properties page from the IBM MQ Explorer, or the `CHANNELS` stanza in the `qm.ini` file, to specify information about channels.

MaxChannels= 100|*number*
The maximum number of *current* channels allowed.

The value must be in the range 1 - 65535. The default is 100.

MaxActiveChannels= *MaxChannels_value*
The maximum number of channels allowed to be *active* at any time. The default is the value specified for the `MaxChannels` attribute.

MaxInitiators= 3|*number*
The maximum number of initiators. The default and maximum value is 3.

MQIBindType=FASTPATH|STANDARD
The binding for applications:

FASTPATH
Channels connect using `MQCONNX` FASTPATH; there is no agent process.

STANDARD
Channels connect using STANDARD.

PipeLineLength=1|*number*
The maximum number of concurrent threads a channel will use. The default is 1. Any value greater than 1 is treated as 2.

When you use pipelining, configure the queue managers at both ends of the channel to have a *PipeLineLength* greater than 1.

Note: Pipelining is only effective for TCP/IP channels.

AdoptNewMCA= NO|SVR|SDR|RCVR|CLUSRCVR|ALL|FASTPATH
If IBM MQ receives a request to start a channel, but finds that an instance of the channel is already running, in some cases the existing channel instance must be stopped before the new one can start. The `AdoptNewMCA` attribute allows you to control which types of channels can be ended in this way.

If you specify the `AdoptNewMCA` attribute for a particular channel type, but the new channel fails to start because a matching channel instance is already running:

1. The new channel tries to stop the previous one by requesting it to end.
2. If the previous channel server does not respond to this request by the time the AdoptNewMCATimeout wait interval expires, the thread or process for the previous channel server is ended.

3. If the previous channel server has not ended after step 2, and after the AdoptNewMCATimeout wait interval expires for a second time, IBM MQ ends the channel with a `CHANNEL IN USE` error.

The AdoptNewMCA functionality applies to server, sender, receiver, and cluster-receiver channels. In the case of a sender or server channel, only one instance of a channel with a particular name can be running in the receiving queue manager. In the case of a receiver or cluster-receiver channel, multiple instances of a channel with a particular name might be running in the receiving queue manager, but only one instance can run at any one time from a particular remote queue manager.

Note: AdoptNewMCA is not supported on requester or server-connection channels.

Specify one or more values, separated by commas or blanks, from the following list:

NO The `AdoptNewMCA` feature is not required. This is the default.

SVR
Adopt server channels.

SDR
Adopt sender channels.

RCVR
Adopt receiver channels.

CLUSRCVR
Adopt cluster receiver channels.

ALL
Adopt all channel types except FASTPATH channels.

FASTPATH
Adopt the channel if it is a FASTPATH channel. This happens only if the appropriate channel type is also specified, for example: `AdoptNewMCA=RCVR,SVR,FASTPATH`.

Attention!: The AdoptNewMCA attribute might behave in an unpredictable fashion with FASTPATH channels. Exercise great caution when enabling the AdoptNewMCA attribute for FASTPATH channels.

AdoptNewMCATimeout= 60|1 - 3600
The amount of time, in seconds, that the new channel instance waits for the old channel instance to end. Specify a value in the range 1 - 3600. The default value is 60.

AdoptNewMCACheck=QM|ADDRESS|NAME|ALL
The type of checking required when enabling the `AdoptNewMCA` attribute. If possible, perform full checking to protect your channels from being shut down, inadvertently or maliciously. At the very least, check that the channel names match.

Specify one or more of the following values, separated by commas or blanks in the case of *QM*, *NAME*, or *ALL*:

QM Check that the queue manager names match.

Note that the queue manager name itself is matched, not the QMID.

ADDRESS
Check the communications source IP address. For example, the TCP/IP address.

Note: Comma separated CONNAME values apply to target addresses and are, therefore, not relevant to this option.

In the case that a multi-instance queue manager fails over from `hosta` to `hostb`, any outbound channels from that queue manager will use the source IP address of `hostb`. If this is different from `hosta`, then AdoptNewMCACheck=*ADDRESS* fails to match.

You can use SSL or TLS with mutual authentication to prevent an attacker from disrupting an existing running channel. Alternatively, use an HACMP type solution with IP-takeover instead of multi-instance queue managers, or use a network load balancer to mask the source IP address.

NAME
Check that the channel names match.

ALL
Check for matching queue manager names, the communications address, and for matching channel names.

The default is `AdoptNewMCACheck=NAME,ADDRESS,QM`.

V9.0.0.1 **ChlauthEarlyAdopt = Y| 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.

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|always|optional
From IBM MQ Version 8.0, set protected passwords in the MQCSP structure, rather than using TLS.

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, see MQCSP password protection.

Related concepts:

"Channel states" on page 993
A channel can be in one of many states at any time. Some states also have substates. From a given state a channel can move into other states.

TCP, LU62, and NETBIOS

Use these queue manager properties pages, or stanzas in the qm.ini file, to specify network protocol configuration parameters. They override the default attributes for channels.

TCP

Use the TCP queue manager properties page from the IBM MQ Explorer, or the TCP stanza in the qm.ini file, to specify Transmission Control Protocol/Internet Protocol (TCP/IP) configuration parameters.

Port= 1414| ***port_number***

The default port number, in decimal notation, for TCP/IP sessions. The *well known* port number for IBM MQ is 1414.

Library1= ***DLLName1*** **(IBM MQ for Windows only)**

The name of the TCP/IP sockets DLL.

The default is WSOCK32.

KeepAlive= NO|YES

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

ListenerBacklog=number

Override the default number of outstanding requests for the TCP/IP listener.

When receiving on TCP/IP, a maximum number of outstanding connection requests is set. This can be considered to be a backlog of requests waiting on the TCP/IP port for the listener to accept the request. The default listener backlog values are shown in Table 110.

Table 110. Default outstanding connection requests (TCP)

Platform	Default ListenerBacklog value
Windows Server	100
Windows Workstation	5
Linux	100
Solaris	100
HP-UX	20
AIX V5.3 or later	100

Note: Some operating systems support a larger value than the default shown. Use this to avoid reaching the connection limit.

Conversely, some operating systems might limit the size of the TCP backlog, so the effective TCP backlog could be smaller than requested here.

If the backlog reaches the values shown in Table 110, the TCP/IP connection is rejected and the channel cannot start. For message channels, this results in the channel going into a RETRY state and retrying the connection at a later time. For client connections, the client receives an MQRC_Q_MGR_NOT_AVAILABLE reason code from MQCONN and retries the connection at a later time.

The following group of properties can be used to control the size of buffers used by TCP/IP. The values are passed directly to the TCP/IP layer of the operating system. Great care should be taken when using these properties. If the values are set incorrectly it can adversely affect the TCP/IP performance. For further information about how this affects performance refer to the TCP/IP documentation for your environment. A value of zero indicates that the operating system will manage the buffer sizes, as opposed to the buffer sizes being fixed by IBM MQ.

Connect_Timeout= 0|number
The number of seconds before an attempt to connect the socket times out. The default value of zero specifies that there is no connect timeout.

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

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

SndBuffSize=number|0
The size in bytes of the TCP/IP send buffer used by the sending end of channels. This stanza value can be overridden by a stanza more specific to the channel type, for example RcvSndBuffSize. If the value is set as zero, the operating system defaults are used. If no value is set, then the IBM MQ default, 32768, is used.

RcvBuffSize=number|0
The size in bytes of the TCP/IP receive buffer used by the receiving end of channels. This stanza value can be overridden by a stanza more specific to the channel type, for example RcvRcvBuffSize. If the value is set as zero, the operating system defaults are used. If no value is set, then the IBM MQ default, 32768, is used.

RcvSndBuffSize=number|0
The size in bytes of the TCP/IP send buffer used by the sender end of a receiver channel. If the value is set as zero, the operating system defaults are used. If no value is set, then the IBM MQ default, 32768, is used.

RcvRcvBuffSize=number|0
The size in bytes of the TCP/IP receive buffer used by the receiving end of a receiver channel. If the value is set as zero, the operating system defaults are used. If no value is set, then the IBM MQ default, 32768, is used.

SvrSndBuffSize=number|0
The size in bytes of the TCP/IP send buffer used by the server end of a client-connection server-connection channel. If the value is set as zero, the operating system defaults are used. If no value is set, then the IBM MQ default, 32768, is used.

SvrRcvBuffSize=number|0
The size in bytes of the TCP/IP receive buffer used by the server end of a client-connection server-connection channel. If the value is set as zero, the operating system defaults are used. If no value is set, then the IBM MQ default, 32768, is used.

Windows LU62 (IBM MQ for Windows only)

Use the `LU6.2` queue manager properties page from the IBM MQ Explorer, or the `LU62` stanza in the qm.ini file, to specify SNA LU 6.2 protocol configuration parameters.

TPName
The TP name to start on the remote site.

Library1= *DLLName 1*
The name of the APPC DLL.

The default value is WCPIC32.

Library2= *DLLName2*
The same as Library1, used if the code is stored in two separate libraries.

The default value is WCPIC32.

Windows **NETBIOS (IBM MQ for Windows only)**

Use the Netbios queue manager properties page from the IBM MQ Explorer, or the NETBIOS stanza in the qm.ini file, to specify NetBIOS protocol configuration parameters.

LocalName= *name*
The name by which this machine is known on the LAN.

AdapterNum= 0| *adapter_number*
The number of the LAN adapter. The default is adapter 0.

NumSess= 1| *number_of_sessions*
The number of sessions to allocate. The default is 1.

NumCmds= 1| *number_of_commands*
The number of commands to allocate. The default is 1.

NumNames= 1| *number_of_names*
The number of names to allocate. The default is 1.

Library1= *DLLName1*
The name of the NetBIOS DLL.

The default value is NETAPI32.

Windows **SPX (IBM MQ for Windows only)**

Use the SPX queue manager properties page from the IBM MQ Explorer, or the SPX stanza in the qm.ini file, to specify SPX protocol configuration parameters.

Socket= 5E86| *socket_number*
The SPX socket number in hexadecimal notation. The default is X'5E86'.

BoardNum= 0| *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 111 on page 949.

Table 111. Default outstanding connection requests (SPX)

Platform	Default ListenerBacklog value
Windows Server	100
Windows Workstation	5

Note: Some operating systems support a larger value than the default shown. Use this to avoid reaching the connection limit.

Conversely, some operating systems might limit the size of the SPX backlog, so the effective SPX backlog could be smaller than requested here.

If the backlog reaches the values shown in Table 111, the SPX connection is rejected and the channel cannot start. For message channels, this results in the channel going into a RETRY state and retrying the connection at a later time. For client connections, the client receives an MQRC_Q_MGR_NOT_AVAILABLE reason code from `MQCONN` and should retry the connection at a later time.

Exit path

Use the `Exits` queue manager properties page from the IBM MQ Explorer, or the `ExitPath` stanza in the qm.ini file to specify the path for user exit programs on the queue manager system.

`ExitsDefaultPath=` ***`string`***
The ExitsDefaultPath attribute specifies the location of:
- 32-bit channel exits for clients
- 32-bit channel exits and data conversion exits for servers
- Unqualified XA switch load files

`ExitsDefaultPath64=` ***`string`***
The ExitsDefaultPath64 attribute specifies the location of:
- 64-bit channel exits for clients
- 64-bit channel exits and data conversion exits for servers
- Unqualified XA switch load files

API exits:

For a server, use the `Exits` queue manager properties page from the IBM MQ Explorer, or the `ApiExitLocal` stanza in the `qm.ini` file to identify API exit routines for a queue manager. For a client modify the `ApiExitLocal` stanza in the mqclient.ini file to identify API exit routines for a queue manager.

On Windows systems, you can also use the amqmdain command to change the entries for API exits. (To identify API exit routines for all queue managers, you use the `ApiExitCommon` and `ApiExitTemplate` stanzas, as described in "API exits" on page 933.)

Note, that for the API exit to work correctly, the message from the server must be sent to the client unconverted. After the API exit has processed the message, the message must then be converted on the client. This, therefore, requires that you have installed all conversion exits on the client.

For a complete description of the attributes for these stanzas, see Configuring API exits.

Queue manager error logs

Use the `Extended` queue manager properties page from the IBM MQ Explorer, or the `QMErrorLog` stanza in the qm.ini file to tailor the operation and contents of queue manager error logs.

Attention: You can use IBM MQ Explorer to make the changes, only if you are using a local queue manager on the Windows platform.

V 9.0.4 **ErrorLogSize=** ***maxsize***

Specifies the size of the queue manager error log which it is copied to the backup. *maxsize* must be in the range 32768 through 2147483648 bytes. If **ErrorLogSize** is not specified, the default value of 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.

Important: From IBM MQ Version 9.0.4, the default size of the **ErrorLogSize** attribute has increased. This is a change from IBM MQ Version 9.0.3.

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:

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
7234 - Number of messages loaded
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
9489 - SVRCONN max instances limit exceeded
9490 - SVRCONN max instances per client limit exceeded
9508 - Cannot connect to queue manager
9524 - Remote queue manager unavailable
9528 - User requested closure of channel
9545 - Disconnect interval expired
9558 - Remote Channel is not available
9637 - Channel is lacking a certificate
9776 - Channel was blocked by user ID
9777 - Channel was blocked by NOACCESS map
9782 - Connection was blocked by address
9999 - Channel program ended abnormally

SuppressMessage= ***msgIds***

Specifies messages that are written to the queue manager error log once only in a specified time interval. If your IBM MQ system is heavily used, with many channels stopping and starting, a large number of information messages are sent to the z/OS console and hardcopy log. The IBM MQ - IMS bridge and buffer manager might also produce a large number of information messages, so suppressing messages prevents you from receiving a number of repeating messages if you require it. The time interval is specified by `SuppressInterval`. *msgIds* contain a comma-separated list of message identifiers from the following:

5211 - Maximum property name length exceeded.

5973 - Distributed publish/subscribe subscription inhibited

5974 - Distributed publish/subscribe publication inhibited

6254 - The system could not dynamically load shared library

7234 - Number of messages loaded

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

9489 - SVRCONN max instances limit exceeded

9490 - SVRCONN max instances per client limit exceeded

9508 - Cannot connect to queue manager

9524 - Remote queue manager unavailable

9528 - User requested closure of channel

9545 - Disconnect interval expired

9558 - Remote Channel is not available

9637 - Channel is lacking a certificate

9776 - Channel was blocked by user ID

9777 - Channel was blocked by NOACCESS map

9782 - Connection was blocked by address

9999 - Channel program ended abnormally

If the same message ID is specified in both `SuppressMessage` and `ExcludeMessage`, the message is excluded.

SuppressInterval= ***length***

Specifies the time interval, in seconds, in which messages specified in `SuppressMessage` are written to the queue manager error log once only. *length* must be in the range 1 through 86400 seconds. If `SuppressInterval` is not specified, the default value of 30 seconds is used.

Diagnostic message services: V 9.0.5

The diagnostic message services options, introduced in IBM MQ Version 9.0.5 as part of the improvements to diagnostic facilities in error logs, enable you to set up various stanzas, so that output can be directed to different components of your system.

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, 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 aree 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+, +MQ_DATA_PATH+.

`FilePrefix`
The prefix of the log files. The default is AMQERR.

`FileSize`
The size at which the log rolls over. The default is 32MB.

`Format` The format of the file. The value can be either *text* (for 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.

Syslog
This service sends any unfiltered messages to `syslog` on UNIX platforms, using the JSON format.

Note that you can only define one Syslog service.
The severity of the message is mapped to the syslog level in the following way:

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*.

Diagnostic message service stanzas

You enable an additional diagnostic message service, 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=<stanzaname>
Name of a stanza

Service= ***type of service***

This attribute defines a service, where the name of the service is not case sensitive, that are being enabled by this stanza.

For example, to enable `syslog` as an additional service, enter the following:

```
Service=syslog
```

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.

You can add the following optional attributes to the stanzas:
- ExcludeMessage
- SuppressMessage
- SuppressInterval
- “Severities” on page 955

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:

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

7234 - Number of messages loaded

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

9489 - SVRCONN max instances limit exceeded

9490 - SVRCONN max instances per client limit exceeded

9508 - Cannot connect to queue manager

9524 - Remote queue manager unavailable

9528 - User requested closure of channel

9545 - Disconnect interval expired

9558 - Remote Channel is not available

9637 - Channel is lacking a certificate

9776 - Channel was blocked by user ID

9777 - Channel was blocked by NOACCESS map

9782 - Connection was blocked by address

9999 - Channel program ended abnormally

SuppressMessage= ***msgIds***

Specifies messages that are written to the queue manager error log once only in a specified time interval. If your IBM MQ system is heavily used, with many channels stopping and starting, a large number of information messages are sent to the z/OS console and hardcopy log. The IBM MQ - IMS bridge and buffer manager might also produce a large number of information messages, so suppressing messages prevents you from receiving a number of repeating messages if you require it. The time interval is specified by `SuppressInterval`. *msgIds* contain a comma-separated list of message identifiers from the following:

5211 - Maximum property name length exceeded.

5973 - Distributed publish/subscribe subscription inhibited

5974 - Distributed publish/subscribe publication inhibited

6254 - The system could not dynamically load shared library

7234 - Number of messages loaded

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

9489 - SVRCONN max instances limit exceeded

9490 - SVRCONN max instances per client limit exceeded

9508 - Cannot connect to queue manager

9524 - Remote queue manager unavailable

9528 - User requested closure of channel

9545 - Disconnect interval expired

9558 - Remote Channel is not available

9637 - Channel is lacking a certificate

9776 - Channel was blocked by user ID

9777 - Channel was blocked by NOACCESS map

9782 - Connection was blocked by address

9999 - Channel program ended abnormally

If the same message ID is specified in both `SuppressMessage` and `ExcludeMessage`, the message is excluded.

IBM i **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 information:

mqrc command

strmqm command

QMErrorLog service: V 9.0.5

IBM MQ Version 9.0.5 introduces improvements to diagnostic facilities in error logs. This set of facilities is called the **`QMErrorLog`** service. The **`QMErrorLog`** service runs continuously and cannot be turned off.

`ErrorLogSize`

The **`ErrorLogSize`** attribute continues to be supported in the QMErrorLog stanza, with the following changes:
- The default size is increased to 32 MB.
- You can use this attribute to lower the value back to its previous size, if you require to do so.

File service

This service sends any unfiltered messages to a file in a similar way to the QMErrorLog service, using the JSON format diagnostic messages specification.

By default, there are three files called `AMQERR01.json`, `AMQERR02.json`, and `AMQERR03.json`. These files will rollover, based on the configured size.

The following attributes are supported in a File Service stanza only:

FilePath
The path to where the log files should be written. The default is the same location as the current 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+`, `+MQ_DATA_PATH+`.

FilePrefix
The prefix of the log files. The default is `AMQERR`.

FileSize
The size at which the log rolls over, as determined by the **ErrorLogSize** attribute.

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
  Severity=E+
  FilePrefix=Worldending

DiagnosticMessages:
  Name=Communications
  Service=File
  Severity=1, W
  FilePrefix=Information
```

See "Attributes of the stanzas" on page 953 for more information.

UNIX Linux

Syslog service

The Syslog service is not available on Windows or IBM i.

Only one Syslog service can be defined, 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 as follows:
- 0 - LOG_INFO
- 10 - LOG_WARNING
- 20 - LOG_ERR
- 30 - LOG_ERR
- 40 - LOG_ALERT
- 50 - LOG_ALERT

The following attribute is supported in the Syslog stanza only:

Ident Defines the **ident** value associated with the syslog entries. The default value is `ibm-mq`.

The following example shows warning messages being sent to Syslog:

```
DiagnosticMessages:
  Name=WarningsToSyslog
  Service=Syslog
  Severity=W +
  FilePrefix=AllWarnings
```

Queue manager default bind type

Use the `Extended` queue manager properties page from the IBM MQ Explorer, or the `Connection` stanza in the qm.ini file to specify the default bind type. Note that you must create a `Connection` stanza if you need one.

DefaultBindType= `SHARED|ISOLATED`

If DefaultBindType is set to ISOLATED, applications and the queue manager run in separate processes, and no resources are shared between them.

If DefaultBindType is set to SHARED, applications and the queue manager run in separate processes, but some resources are shared between them.

The default is SHARED.

SSL stanza of the queue manager configuration file

Use the SSL stanza of the queue manager configuration file to configure TLS channels on your queue manager.

Online Certificate Status Protocol (OCSP)

A certificate can contain an AuthorityInfoAccess extension. This extension specifies a server to be contacted through Online Certificate Status Protocol (OCSP). To allow SSL or TLS channels on your queue manager to use AuthorityInfoAccess extensions, ensure that the OCSP server named in them is available, is correctly configured, and is accessible over the network. For more information, see Working with revoked certificates.

CrlDistributionPoint (CDP)

A certificate can contain a CrlDistributionPoint extension. This extension contains a URL which identifies both the protocol used to download a certificate revocation list (CRL) and also the server to be contacted.

If you want to allow SSL or TLS channels on your queue manager to use CrlDistributionPoint extensions, ensure that the CDP server named in them is available, correctly configured, and accessible over the network.

The SSL Stanza

Use the SSL stanza in the `qm.ini` file to configure how TLS channels on your queue manager attempts to use the following facilities, and how they react if problems occur when using them.

In each of the following cases, if the value supplied is not one of the valid values listed, then the default value is taken. No error messages are written mentioning that an invalid value is specified.

`CDPCheckExtensions=YES|NO`

CDPCheckExtensions specifies whether 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`: TLS channels do not try to check CDP servers. This value is the default.

`OCSPAuthentication=REQUIRED|WARN|OPTIONAL`

OCSPAuthentication specifies the action to be taken when a revocation status cannot be determined from an OCSP server.

If OCSP checking is enabled, 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`: Failure to determine the revocation status causes the connection to be closed with an error. This value is the default.
- `WARN`: Failure to determine the revocation status causes a warning message to be written in the queue manager error log, but the connection is allowed to proceed.
- `OPTIONAL`: Failure to determine the revocation status allows the connection to proceed silently. No warnings or errors are given.

`OCSPCheckExtensions= YES|NO`

OCSPCheckExtensions specifies whether TLS channels on this queue manager try to check OCSP servers that are named in AuthorityInfoAccess certificate extensions.

- `YES`: 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.

`SSLHTTPProxyName=` ***`string`***

The string is either the host name or network address of the HTTP Proxy server that is to be used by GSKit for OCSP checks. This address can be followed by an optional port number, enclosed in parentheses. If you do not specify the port number, the default HTTP port, 80, is used. On the HP-UX PA-RISC and Sun Solaris SPARC platforms, and for 32-bit clients on AIX, the network address can only be an IPv4 address; on other platforms it can be an IPv4 or IPv6 address.

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

Exit properties

Use the Cluster queue manager properties page from the IBM MQ Explorer, or the ExitPropertiesLocal stanza in the qm.ini file, to specify information about exit properties on a queue manager. Alternatively, you can set it using the **`amqmdain`** command.

By default, this setting is inherited from the CLWLMode attribute in the ExitProperties stanza of the machine-wide configuration (described in "Exit properties" on page 929). Change this setting only if you want to configure this queue manager in a different way. This value can be overridden for individual queue managers using the cluster workload mode attribute on the Cluster queue manager properties page.

`CLWLMode= SAFE|FAST`

The cluster workload (CLWL) exit allows you to specify which cluster queue in the cluster to open in response to an MQI call (for example, `MQOPEN`, `MQPUT`). The CLWL exit runs either in FAST mode or SAFE mode depending on the value you specify on the CLWLMode attribute. If you omit the CLWLMode attribute, the cluster workload exit runs in SAFE mode.

`SAFE`

Run the CLWL exit in a separate process from the queue manager. This is the default.

If a problem arises with the user-written CLWL exit when running in SAFE mode, the following happens:

- The CLWL server process (amqzlwa0) fails.
- The queue manager restarts the CLWL server process.
- The error is reported to you in the error log. If an MQI call is in progress, you receive notification in the form of a return code.

The integrity of the queue manager is preserved.

Note: Running the CLWL exit in a separate process can affect performance.

`FAST`

Run the cluster exit inline in the queue manager process.

Specifying this option improves performance by avoiding the process switching costs associated with running in SAFE mode, but does so at the expense of queue manager integrity. You should only run the CLWL exit in FAST mode if you are convinced that there are **no** problems with your CLWL exit, and you are particularly concerned about performance.

If a problem arises when the CLWL exit is running in FAST mode, the queue manager will fail and you run the risk of the integrity of the queue manager being compromised.

Subpool

This stanza is created by IBM MQ. Do not change it.

The stanza Subpool, and attribute ShortSubpoolName within that stanza, are written automatically by IBM MQ when you create a queue manager. IBM MQ chooses a value for ShortSubpoolName. Do not alter this value.

The name corresponds to a directory and symbolic link created inside the /var/mqm/sockets directory, which IBM MQ uses for internal communications between its running processes.

Filesystem

From IBM WebSphere MQ Version 7.0 onwards, directory and file systems permissions are more restricted. By default, only members of the `mqm` group can write directly to error log files and First Failure Data Capture files. You can use the **`Filesystem`** stanza to allow users who are not members of the `mqm` group to access error directories and files.

To allow users who are not members of the `mqm` group or, on IBM i, are not members of the QMQMADM group, to access error directories and files, you must set:

`ValidateAuth=`
    `No`

Note that the text is case sensitive.

IBM i On IBM i, you must also set the authority for the additional users to *PUBLIC.

Note: IBM MQ does not support the addition of users to error logs.

You can use this to extend access, by changing the group ownership of the directory and using `setgid` permissions. For example, to widen access to include members of a group called `mqerrors`, use the following:

```
chgrp mqerrors /var/mqm/errors
chgrp mqerrors /var/mqm/qmgrs/QMname/errors
chmod 6770 /var/mqm/qmgrs/QMname/errors
```

This causes all files within these directories to be created with `mqerrors` ownership, rather than `mqm` ownership. Hence, extending access to the members of the `mqerrors` group.

This approach does not provide o+r permissions on the actual files. Alternatively, a cron job (running under `mqm`) could periodically change the permissions of the files within these directories, to provide o+r permissions

Security

Use the `Security` stanza in the `qm.ini` file to specify options for the Object Authority Manager (OAM).

`ClusterQueueAccessControl=RQMName|Xmitq`
Set this attribute to check the access control of cluster queues or fully qualified queues hosted on cluster queue managers.

`RQMName`
The profiles checked for access control of remotely hosted queues are named queues or named queue manager profiles.

`Xmitq`
The profiles checked for access control of remotely hosted queues are resolved to the `SYSTEM.CLUSTER.TRANSMIT.QUEUE`.

`Xmitq` is the default value.

`GroupModel=GlobalGroups`
This attribute determines whether the OAM checks global groups when determining the group membership of a user on Windows.

The default is not to check global groups.

`GlobalGroups`
The OAM checks global groups.

With `GlobalGroups` set, the authorization commands, **`setmqaut`**, **`dspmqaut`**, and **`dmpmqaut`** accept global groups names; see the **`setmqaut -g`** parameter.

Note: Setting the `ClusterQueueAcessControl=RQMName` and having a custom implementation of the Authorization Service at less than `MQZAS_VERSION_6` results in the queue manager not starting. In this instance, either set `ClusterQueueAcessControl=Xmitq` or upgrade the custom Authorization Service to `MQZAS_VERSION_6` or greater.

Tuning parameters

V 9.0.5

Use the `TuningParameters` stanza in the `qm.ini` file to specify options for tuning the queue manager.

`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.

Related information:
Implicit syncpoint

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.

Procedure

Use the information in the following subtopics to connect your applications using distributed queuing:
- "IBM MQ distributed queuing techniques"
- "Introduction to distributed queue management" on page 982
- "How to send a message to another queue manager" on page 985
- "Triggering channels" on page 1008
- "Safety of messages" on page 1005
- ULW "Monitoring and controlling channels on UNIX, Linux, and Windows" on page 1015
- IBM i "Monitoring and controlling channels on IBM i" on page 1039

Related concepts:

z/OS "Customizing IBM MQ for z/OS" on page 1459
Use this topic as a step by step guide for customizing your IBM MQ system.

z/OS "Setting up communications with other queue managers" on page 1531
This section describes the IBM MQ for z/OS preparations you need to make before you can start to use distributed queuing.

Related tasks:

"Configuring connections between the server and client" on page 844
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 1063
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 and queue manager configuration information" on page 906
You can change the behavior of IBM MQ or an individual queue manager to suit the needs of your installation.

z/OS "Configuring queue managers on z/OS" on page 1455
Use these instructions to configure queue managers on IBM MQ for z/OS.

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:

- ULW Message channel planning example for UNIX, Linux, and Windows
- IBM i Message channel planning example for IBM i
- z/OS Message channel planning example for z/OS
- z/OS Message channel planning example for z/OS using queue-sharing groups

Related tasks:

"Configuring distributed queuing" on page 961
This section provides more detailed information about intercommunication between IBM MQ installations, including queue definition, channel definition, triggering, and sync point procedures

Related information:

Channels

Introduction to message queuing

Distributed queuing and clusters

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 961. If your queue manager is in a cluster, message flow is controlled using different techniques, as described in "Message flow control." z/OS If your queue managers are in a queue-sharing group and intra-group queuing (IGQ) is enabled, then the message flow can be controlled by IGQ agents. These agents are described in Intra-group queuing.

You can use the following objects to achieve message flow control:
- Transmission queues
- Message channels
- Remote queue definition
- Queue manager alias definition
- Reply-to queue alias definition

The queue manager and queue objects are described in Object types. Message channels are described in Distributed queuing components. The following techniques use these objects to create message flows in your system:
- Putting messages to remote queues
- Routing by way of particular transmission queues
- Receiving messages
- Passing messages through your system
- Separating message flows
- Switching a message flow to another destination
- Resolving the reply-to queue name to an alias name

Note

All the concepts described in this section are relevant for all nodes in a network, and include sending and receiving ends of message channels. For this reason, only one node is illustrated in most examples. The exception is where the example requires explicit cooperation by the administrator at the other end of a message channel.

Before proceeding to the individual techniques, it is useful to recap on the concepts of name resolution and the three ways of using remote queue definitions. See Distributed queuing and clusters.

Related concepts:

"Queue names in transmission header"
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"
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 112 on page 964 explains how to define each of the three ways:

- Using a remote queue definition to redefine a local queue name.

  The application provides only the queue name when opening a queue, and this queue name is the name of the remote queue definition.

  The remote queue definition contains the names of the target queue and queue manager. Optionally, the definition can contain the name of the transmission queue to be used. If no transmission queue name is provided, the queue manager uses the queue manager name, taken from the remote queue definition, for the transmission queue name. If a transmission queue of this name is not defined, but a default transmission queue is defined, the default transmission queue is used.
- Using a remote queue definition to redefine a queue manager name.

  The application, or channel program, provides a queue name together with the remote queue manager name when opening the queue.

  If you have provided a remote queue definition with the same name as the queue manager name, and you have left the queue name in the definition blank, then the queue manager substitutes the queue manager name in the open call with the queue manager name in the definition.

  In addition, the definition can contain the name of the transmission queue to be used. If no transmission queue name is provided, the queue manager takes the queue manager name, taken from the remote queue definition, for the transmission queue name. If a transmission queue of this name is not defined, but a default transmission queue is defined, the default transmission queue is used.
- Using a remote queue definition to redefine a reply-to queue name.

  Each time an application puts a message to a queue, it can provide the name of a reply-to queue for answer messages but with the queue manager name blank.

If you provide a remote queue definition with the same name as the reply-to queue then the local queue manager replaces the reply-to queue name with the queue name from your definition.

You can provide a queue manager name in the definition, but not a transmission queue name.

Table 112. Three ways of using the remote queue definition object

Usage	**Queue manager name**	**Queue name**	**Transmission queue name**
1. Remote queue definition (on OPEN call)			
Supplied in the call	blank or local QM	(*) required	not applicable
Supplied in the definition	required	required	optional
2. Queue manager alias (on OPEN call)			
Supplied in the call	(*) required and not local QM	required	not applicable
Supplied in the definition	required	blank	optional
3. Reply-to queue alias (on PUT call)			
Supplied in the call	blank	(*) required	not applicable
Supplied in the definition	optional	optional	blank

Note: (*) means that this name is the name of the definition object

For a formal description, see Queue name resolution.

Putting messages on remote queues

You can use remote queue definition objects to resolve a queue name to a transmission queue to an adjacent queue manager.

In a distributed-queuing environment, a transmission queue and channel are the focal point for all messages to a location whether the messages originate from applications in your local system, or arrive through channels from an adjacent system. Figure 84 on page 965 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.

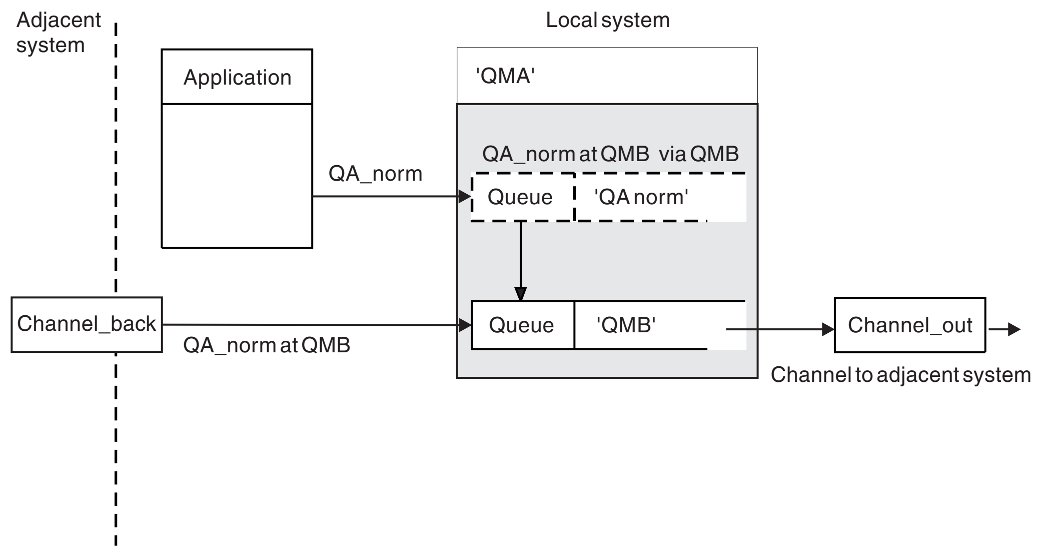


Figure 84. A remote queue definition is used to resolve a queue name to a transmission queue to an adjacent queue manager. Note: The dashed outline represents a remote queue definition. This queue is not a real queue, but a name alias that is controlled as though it were a real queue.

If you are the IBM MQ system administrator, you must:

- Define the message channel from the adjacent system
- Define the message channel to the adjacent system
- Create the transmission queue QMB
- Define the remote queue object 'QA_norm' to resolve the queue name used by applications to the destination queue name, destination queue manager name, and transmission queue name

In a clustering environment, you only need to define a cluster-receiver channel at the local queue manager. You do not need to define a transmission queue or a remote queue object. See Clusters.

More about name resolution

The effect of the remote queue definition is to define a physical destination queue name and queue manager name. These names are put in the transmission headers of messages.

Incoming messages from an adjacent system have already had this type of name resolution carried out by the original queue manager. Therefore they have the transmission header showing the physical destination queue name and queue manager name. These messages are unaffected by remote queue definitions.

Choosing the transmission queue

You can use a remote queue definition to allow a different transmission queue to send messages to the same adjacent queue manager.

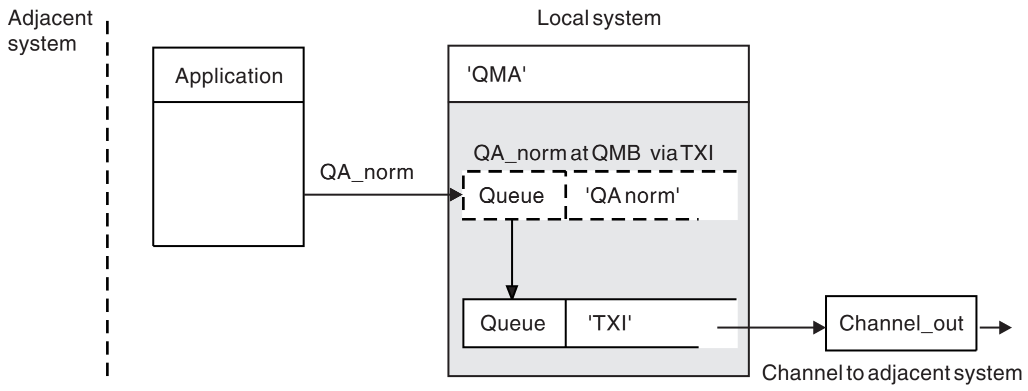


Figure 85. The remote queue definition allows a different transmission queue to be used

In a distributed-queuing environment, when you need to change a message flow from one channel to another, use the same system configuration as shown in Figure 84 on page 965 in “Putting messages on remote queues” on page 964. Figure 85 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 85, you must provide the remote queue object 'QA_norm', and the transmission queue 'TX1'. You must provide 'QA_norm' to choose the 'QA_norm' queue at the remote queue manager, the transmission queue 'TX1', and the queue manager 'QMB_priority'. Specify 'TX1' in the definition of the channel adjacent to the system.

Messages are placed on transmission queue 'TX1' with a transmission header containing 'QA_norm at QMB_priority', and are sent over the channel to the adjacent system.

The channel_back has been left out of this illustration because it would need a queue manager alias.

In a clustering environment, you do not need to define a transmission queue or a remote queue definition. For more information, see “Defining cluster queues” on page 1064.

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.

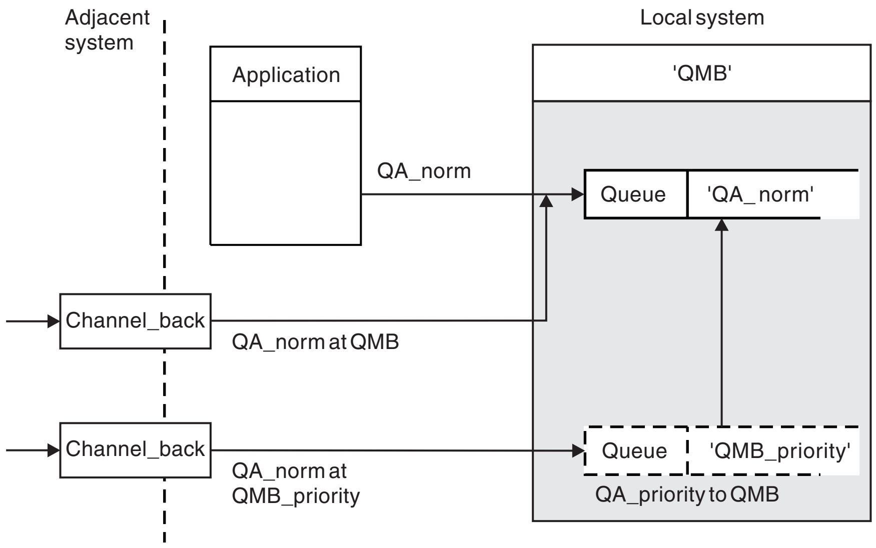


Figure 86. Receiving messages directly, and resolving alias queue manager name

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 86 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.

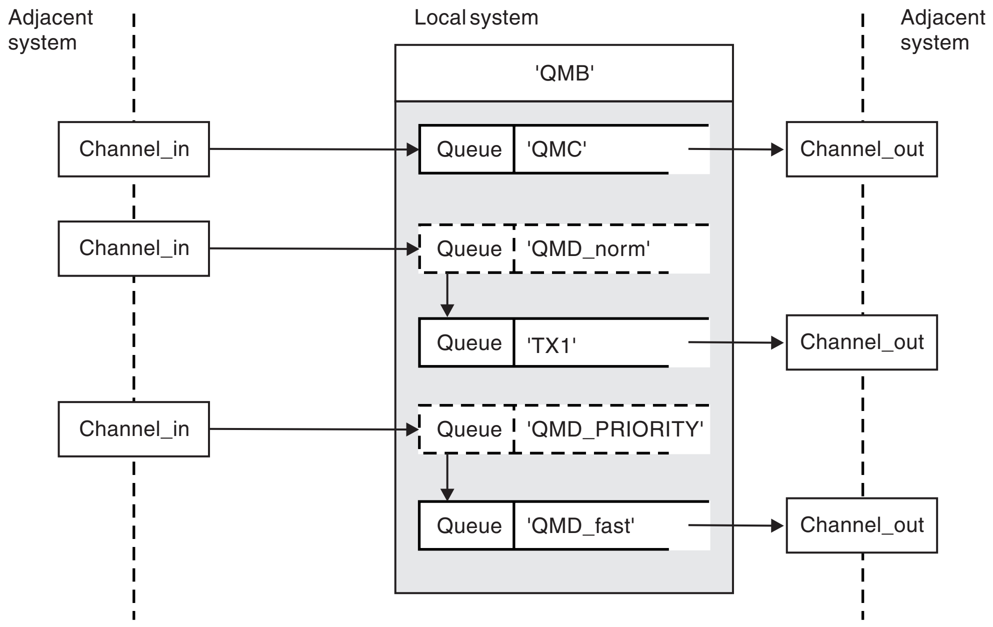


Figure 87. Three methods of passing messages through your system

The technique shown in Figure 86 on page 967 in “Receiving messages” on page 967, showed how an alias flow is captured. Figure 87 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.

In a distributed-queuing environment, the need to separate messages to the same queue manager into different message flows can arise for a number of reasons. For example:

- You might need to provide a separate flow for large, medium, and small messages. This need also applies in a clustering environment and, in this case, you can create clusters that overlap. There are a number of reasons you might do so, for example:
  - To allow different organizations to have their own administration.
  - To allow independent applications to be administered separately.
  - To create a class of service. For example, you could have a cluster called STAFF that is a subset of the cluster called STUDENTS. When you put a message to a queue advertised in the STAFF cluster, a restricted channel is used. When you put a message to a queue advertised in the STUDENTS cluster, either a general channel or a restricted channel can be used.
  - To create test and production environments.
- It might be necessary to route incoming messages by different paths from the path of the locally generated messages.
- Your installation might require to schedule the movement of messages at certain times (for example, overnight) and the messages then need to be stored in reserved queues until scheduled.

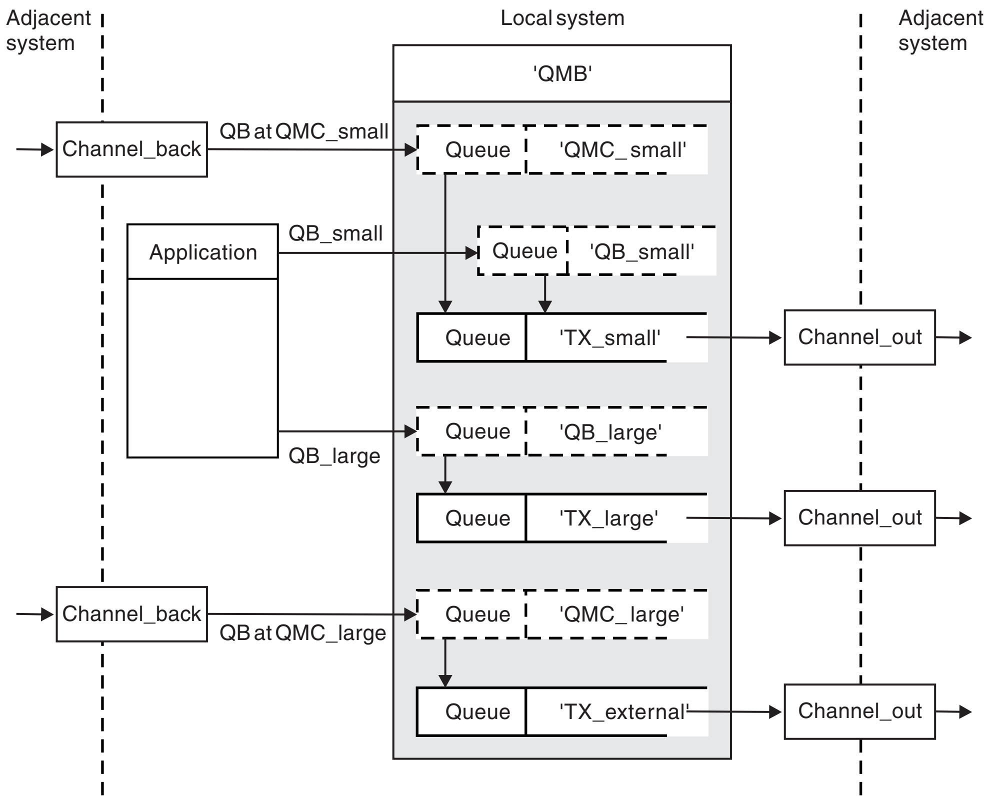


Figure 88. Separating messages flows

In the example shown in Figure 88, 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.

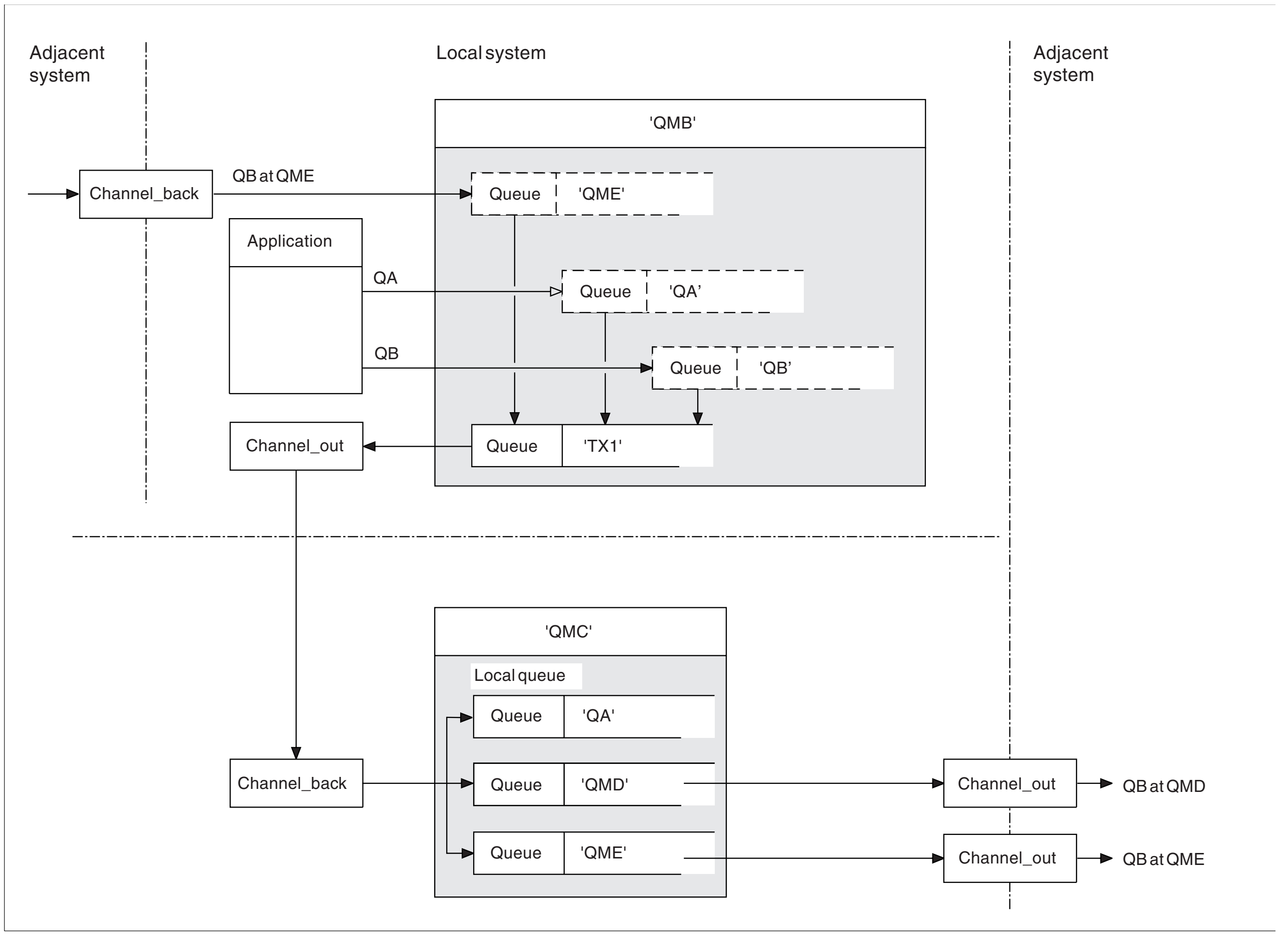


Figure 89. Combining message flows on to a channel

Figure 89 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.

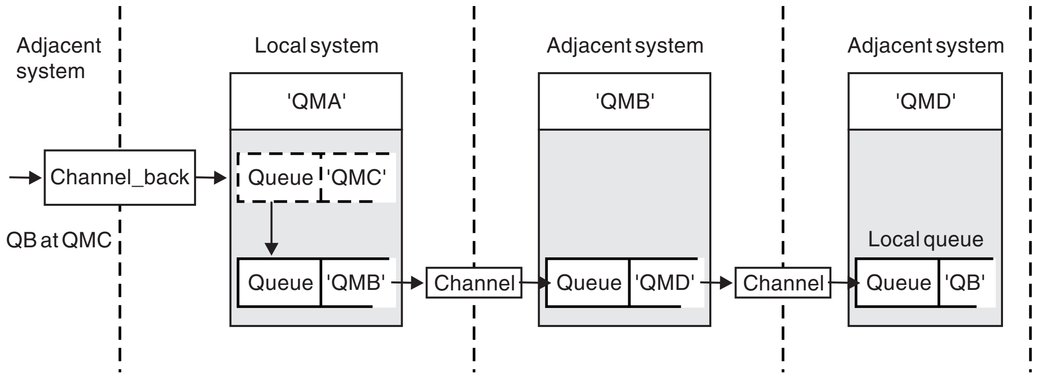


Figure 90. Diverting message streams to another destination

Figure 90 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 1145.

Sending messages to a distribution list

You can use a single MQPUT call to have an application send a message to several destinations.

In IBM MQ on all platforms except z/OS, an application can send a message to several destinations with a single MQPUT call. You can do so in both a distributed-queuing environment and a clustering environment. You have to define the destinations in a distribution list, as described in Distribution lists.

Not all queue managers support distribution lists. When an MCA establishes a connection with a partner, it determines whether the partner supports distribution lists and sets a flag on the transmission queue accordingly. If an application tries to send a message that is destined for a distribution list but the partner does not support distribution lists, the sending MCA intercepts the message and puts it onto the transmission queue once for each intended destination.

A receiving MCA ensures that messages sent to a distribution list are safely received at all the intended destinations. If any destinations fail, the MCA establishes which ones have failed. It then can generate exception reports for them and can try to send the messages to them again.

Reply-to queue

You can create a complete remote queue processing loop using a reply-to queue.

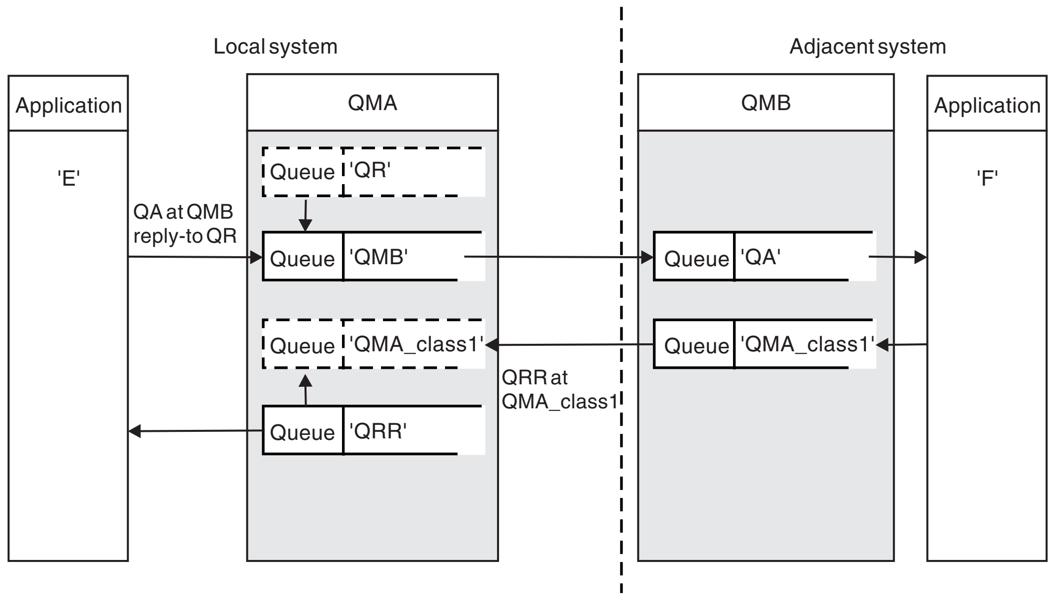


Figure 91. Reply-to queue name substitution during PUT call

A complete remote queue processing loop using a reply-to queue is shown in Figure 91. This loop applies in both a distributed-queuing environment and a clustering environment. The details are as shown in Table 116 on page 981.

The application opens QA at QMB and puts messages on that queue. The messages are given a reply-to queue name of QR, without the queue manager name being specified. Queue manager QMA finds the reply-to queue object QR and extracts from it the alias name of QRR and the queue manager name QMA_class1. These names are put into the reply-to fields of the messages.

Reply messages from applications at QMB are addressed to QRR at QMA_class1. The queue manager alias name definition QMA_class1 is used by the queue manager to flow the messages to itself, and to queue QRR.

This scenario depicts the way you give applications the facility to choose a class of service for reply messages. The class is implemented by the transmission queue QMA_class1 at QMB, together with the queue manager alias definition, QMA_class1 at QMA. In this way, you can change an application's reply-to queue so that the flows are segregated without involving the application. The application always chooses QR for this particular class of service. You have the opportunity to change the class of service with the reply-to queue definition QR.

You must create:
- Reply-to queue definition QR
- Transmission queue object QMB
- Channel_out definition
- Channel_back definition
- Queue manager alias definition QMA_class1
- Local queue object QRR, if it does not exist

The complementary administrator at the adjacent system must create:
- Receiving channel definition
- Transmission queue object QMA_class1
- Associated sending channel
- Local queue object QA.

Your application programs use:
- Reply-to queue name QR in put calls
- Queue name QRR in get calls

In this way, you can change the class of service as necessary, without involving the application. You change the reply-to alias 'QR', together with the transmission queue 'QMA_class1' and queue manager alias 'QMA_class1'.

If no reply-to alias object is found when the message is put on the queue, the local queue manager name is inserted in the blank reply-to queue manager name field. The reply-to queue name remains unchanged.

Name resolution restriction

Because the name resolution has been carried out for the reply-to queue at 'QMA' when the original message was put, no further name resolution is allowed at 'QMB'. The message is put with the physical name of the reply-to queue by the replying application.

The applications must be aware that the name they use for the reply-to queue is different from the name of the actual queue where the return messages are to be found.

For example, when two classes of service are provided for the use of applications with reply-to queue alias names of 'C1_alias', and 'C2_alias', the applications use these names as reply-to queue names in the message put calls. However, the applications actually expect messages to appear in queues 'C1' for 'C1_alias' and 'C2' for 'C2_alias'.

However, an application is able to make an inquiry call on the reply-to alias queue to check for itself the name of the real queue it must use to get the reply messages.

Related concepts:

"How to create queue manager and reply-to aliases" on page 963
This topic explains the three ways that you can create a remote queue definition.

"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.

"How the example works" on page 977
An explanation of the example and how the queue manager uses the reply-to queue alias.

"Reply-to queue alias walk-through" on page 977
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 92, the return route must be available for the reply messages, including the transmission queue, channel, and queue manager alias.

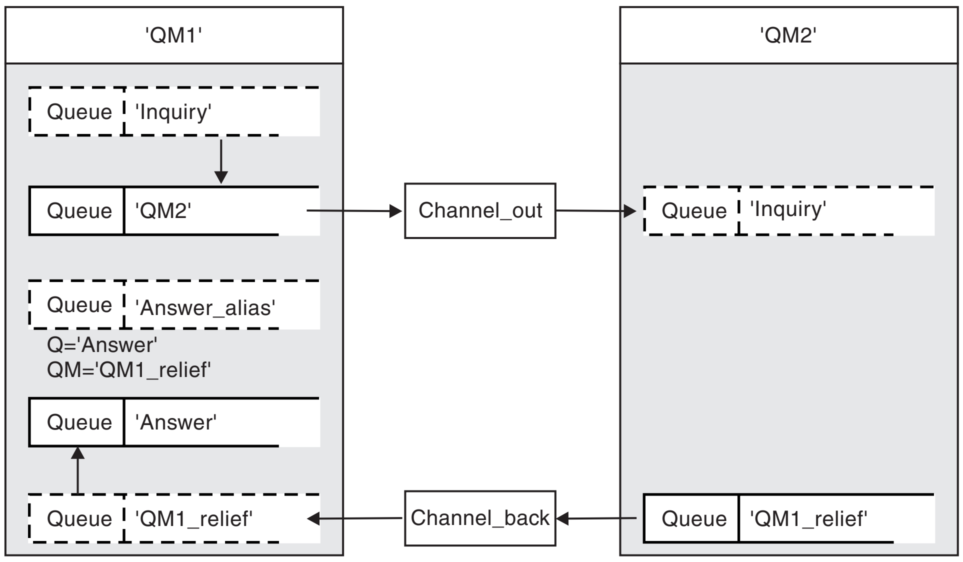


Figure 92. Reply-to queue alias example

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)
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 113. Reply-to queue alias

Field name	Put call	Transmission header
Reply-to queue name	Answer_alias	Answer
Reply-to queue manager name	(blank)	QM1_relief

Reply-to queue alias walk-through:

A walk-through of the process from an application putting a message on a remote queue through to the same application removing the reply message from the alias reply-to queue.

To complete this example, let us look at the process.

1. The application opens a queue named 'Inquiry', and puts messages to it. The application sets the reply-to fields of the message descriptor to:

Reply-to queue name	**Answer_alias**
Reply-to queue manager name	(blank)

2. Queue manager 'QM1' responds to the blank queue manager name by checking for a remote queue definition with the name 'Answer_alias'. If none is found, the queue manager places its own name, 'QM1', in the reply-to queue manager field of the message descriptor.
3. If the queue manager finds a remote queue definition with the name 'Answer_alias', it extracts the queue name and queue manager names from the definition (queue name='Answer' and queue manager name= 'QM1_relief'). It then puts them into the reply-to fields of the message descriptor.
4. The queue manager 'QM1' uses the remote queue definition 'Inquiry' to determine that the intended destination queue is at queue manager 'QM2', and the message is placed on the transmission queue 'QM2'. 'QM2' is the default transmission queue name for messages destined for queues at queue manager 'QM2'.
5. When queue manager 'QM1' puts the message on the transmission queue, it adds a transmission header to the message. This header contains the name of the destination queue, 'Inquiry', and the destination queue manager, 'QM2'.
6. The message arrives at queue manager 'QM2', and is placed on the 'Inquiry' local queue.
7. An application gets the message from this queue and processes the message. The application prepares a reply message, and puts this reply message on the reply-to queue name from the message descriptor of the original message:

Reply-to queue name	**Answer**
Reply-to queue manager name	QM1_relief

8. Queue manager 'QM2' carries out the put command. Finding that the queue manager name, 'QM1_relief', is a remote queue manager, it places the message on the transmission queue with the same name, 'QM1_relief'. The message is given a transmission header containing the name of the destination queue, 'Answer', and the destination queue manager, 'QM1_relief'.
9. The message is transferred to queue manager 'QM1'. The queue manager, recognizes that the queue manager name 'QM1_relief' is an alias, extracts from the alias definition 'QM1_relief' the physical queue manager name 'QM1'.
10. Queue manager 'QM1' then puts the message on the queue name contained in the transmission header, 'Answer'.
11. The application extracts its reply message from the queue 'Answer'.

Networking considerations

In a distributed-queuing environment, because message destinations are addressed with just a queue name and a queue manager name, certain rules apply.

1. Where the queue manager name is given, and the name is different from the local queue manager name:
   - A transmission queue must be available with the same name. This transmission queue must be part of a message channel moving messages to another queue manager, or
   - A queue manager alias definition must exist to resolve the queue manager name to the same, or another queue manager name, and optional transmission queue, or
   - If the transmission queue name cannot be resolved, and a default transmission queue has been defined, the default transmission queue is used.
2. Where only the queue name is supplied, a queue of any type but with the same name must be available on the local queue manager. This queue can be a remote queue definition which resolves to: a transmission queue to an adjacent queue manager, a queue manager name, and an optional transmission queue.

To see how this works in a clustering environment, see Clusters.

z/OS If the queue managers are running in a queue-sharing group (QSG) and intra-group queuing (IGQ) is enabled, you can use the SYSTEM.QSG.TRANSMIT.QUEUE. For more information, see Intra-group queuing.

Consider the scenario of a message channel moving messages from one queue manager to another in a distributed-queuing environment.

The messages being moved have originated from any other queue manager in the network, and some messages might arrive that have an unknown queue manager name as destination. This issue can occur when a queue manager name has changed or has been removed from the system, for example.

The channel program recognizes this situation when it cannot find a transmission queue for these messages, and places the messages on your undelivered-message (dead-letter) queue. It is your responsibility to look for these messages and arrange for them to be forwarded to the correct destination. Alternatively, return them to the originator, where the originator can be ascertained.

Exception reports are generated in these circumstances, if report messages were requested in the original message.

Name resolution convention

Name resolution that changes the identity of the destination queue (that is, logical to physical name changing), only occurs once, and only at the originating queue manager.

Subsequent use of the various alias possibilities must only be used when separating and combining message flows.

Return routing

Messages can contain a return address in the form of the name of a queue and queue manager. This return address form can be used in both a distributed-queuing environment and a clustering environment.

This address is normally specified by the application that creates the message. It can be modified by any application that then handles the message, including user exit applications.

Irrespective of the source of this address, any application handling the message might choose to use this address for returning answer, status, or report messages to the originating application.

The way these response messages are routed is not different from the way the original message is routed. You need to be aware that the message flows you create to other queue managers need corresponding return flows.

Physical name conflicts

The destination reply-to queue name has been resolved to a physical queue name at the original queue manager. It must not be resolved again at the responding queue manager.

It is a likely possibility for name conflict problems that can only be prevented by a network-wide agreement on physical and logical queue names.

Managing queue name translations

When you create a queue manager alias definition or a remote queue definition, the name resolution is carried out for every message carrying that name. This situation must be managed.

This description is provided for application designers and channel planners concerned with an individual system that has message channels to adjacent systems. It takes a local view of channel planning and control.

When you create a queue manager alias definition or a remote queue definition, the name resolution is carried out for every message carrying that name, regardless of the source of the message. To oversee this situation, which might involve large numbers of queues in a queue manager network, you keep tables of:

- The names of source queues and of source queue managers with respect to resolved queue names, resolved queue manager names, and resolved transmission queue names, with method of resolution
- The names of source queues with respect to:
  - Resolved destination queue names
  - Resolved destination queue manager names
  - Transmission queues
  - Message channel names
  - Adjacent system names
  - Reply-to queue names

Note: The use of the term *source* in this context refers to the queue name or the queue manager name provided by the application, or a channel program when opening a queue for putting messages.

An example of each of these tables is shown in Table 114, Table 115, and Table 116 on page 981.

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 114. 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 115. 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

Table 115. Queue name resolution at queue manager QMB (continued)

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
(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 116. 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 assure that messages are delivered, delivered without duplication, and stored in the same order as they were taken from the transmission queue.

The sequence number is generated at the sending end of the channel and is incremented by one before being used, which means that the current sequence number is the number of the last message sent. This information can be displayed using DISPLAY CHSTATUS. The sequence number and an identifier called the LUWID are stored in persistent storage for the last message transferred in a batch. These values are used during channel start-up to ensure that both ends of the link agree on which messages have been transferred successfully.

Sequential retrieval of messages

If an application puts a sequence of messages to the same destination queue, those messages can be retrieved in sequence by a ***single*** application with a sequence of MQGET operations, if the following conditions are met:

- All the put requests were done from the same application.
- All the put requests were either from the same unit of work, or all the put requests were made outside of a unit of work.
- The messages all have the same priority.
- The messages all have the same persistence.
- For remote queuing, the configuration is such that there can only be one path from the application making the put request, through its queue manager, through intercommunication, to the destination queue manager and the target queue.
- The messages are not put to a dead-letter queue (for example, if a queue is temporarily full).
- The application getting the message does not deliberately change the order of retrieval, for example by specifying a particular `MsgId` or `CorrelId` or by using message priorities.
- Only one application is doing get operations to retrieve the messages from the destination queue. If there is more than one application, these applications must be designed to get all the messages in each sequence put by a sending application.

Note: Messages from other tasks and units of work might be interspersed with the sequence, even where the sequence was put from within a single unit of work.

If these conditions cannot be met, and the order of messages on the target queue is important, then the application can be coded to use its own message sequence number as part of the message to assure the order of the messages.

Sequence of retrieval of fast, nonpersistent messages

Nonpersistent messages on a fast channel might overtake persistent messages on the same channel and so arrive out of sequence. The receiving MCA puts the nonpersistent messages on the destination queue immediately and makes them visible. Persistent messages are not made visible until the next sync point.

Loopback testing

Loopback testing is a technique on non- z/OS platforms that allows you to test a communications link without actually linking to another machine.

You set up a connection between two queue managers as though they are on separate machines, but you test the connection by looping back to another process on the same machine. This technique means that you can test your communications code without requiring an active network.

The way you do so depends on which products and protocols you are using.

On Windows systems, you can use the "loopback" adapter.

Refer to the documentation for the products you are using for more information.

Route tracing and activity recording

You can confirm the route a message takes through a series of queue managers in two ways.

You can use the IBM MQ display route application, available through the control command **dspmqrte**, or you can use activity recording. Both of these topics are described in Monitoring reference.

Introduction to distributed queue management

Distributed queue management (DQM) is used to define and control communication between queue managers.

Distributed queue management:

- Enables you to define and control communication channels between queue managers
- Provides you with a message channel service to move messages from a type of *local queue*, known as a transmission queue, to communication links on a local system, and from communication links to local queues at a destination queue manager
- Provides you with facilities for monitoring the operation of channels and diagnosing problems, using panels, commands, and programs

Channel definitions associate channel names with transmission queues, communication link identifiers, and channel attributes. Channel definitions are implemented in different ways on different platforms. Message sending and receiving is controlled by programs known as *message channel agents* (MCAs), which use the channel definitions to start 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:

- ULW "Monitoring and controlling channels on UNIX, Linux, and Windows" on page 1015
- z/OS "Monitoring and controlling channels on z/OS" on page 1534
- IBM i "Monitoring and controlling channels on IBM i" on page 1039

For message channel planning examples, see:

- ULW Message channel planning example for UNIX, Linux, and Windows
- IBM i Message channel planning example for IBM i
- z/OS Message channel planning example for z/OS
- z/OS Message channel planning example for z/OS using queue-sharing groups

For information about channel exits, see Channel-exit programs for messaging channels.

Related concepts:

"Message sending and receiving"
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 991
The channel control function provides facilities for you to define, monitor, and control channels.

"What happens when a message cannot be delivered?" on page 1006
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 1011
The handling of channel initialization data depends on your IBM MQ platform.

"Data conversion" on page 1012
IBM MQ messages might require data conversion when sent between queues on different queue managers.

"Writing your own message channel agents" on page 1013
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 1014
Other topics to consider when preparing IBM MQ for distributed queue management. This topic covers Undelivered-message queue, Queues in use, System extensions and user-exit programs, and Running channels and listeners as trusted applications.

Related information:

Example configuration information

Message sending and receiving

The following figure shows the distributed queue management model, detailing the relationships between entities when messages are transmitted. It also shows the flow for control.

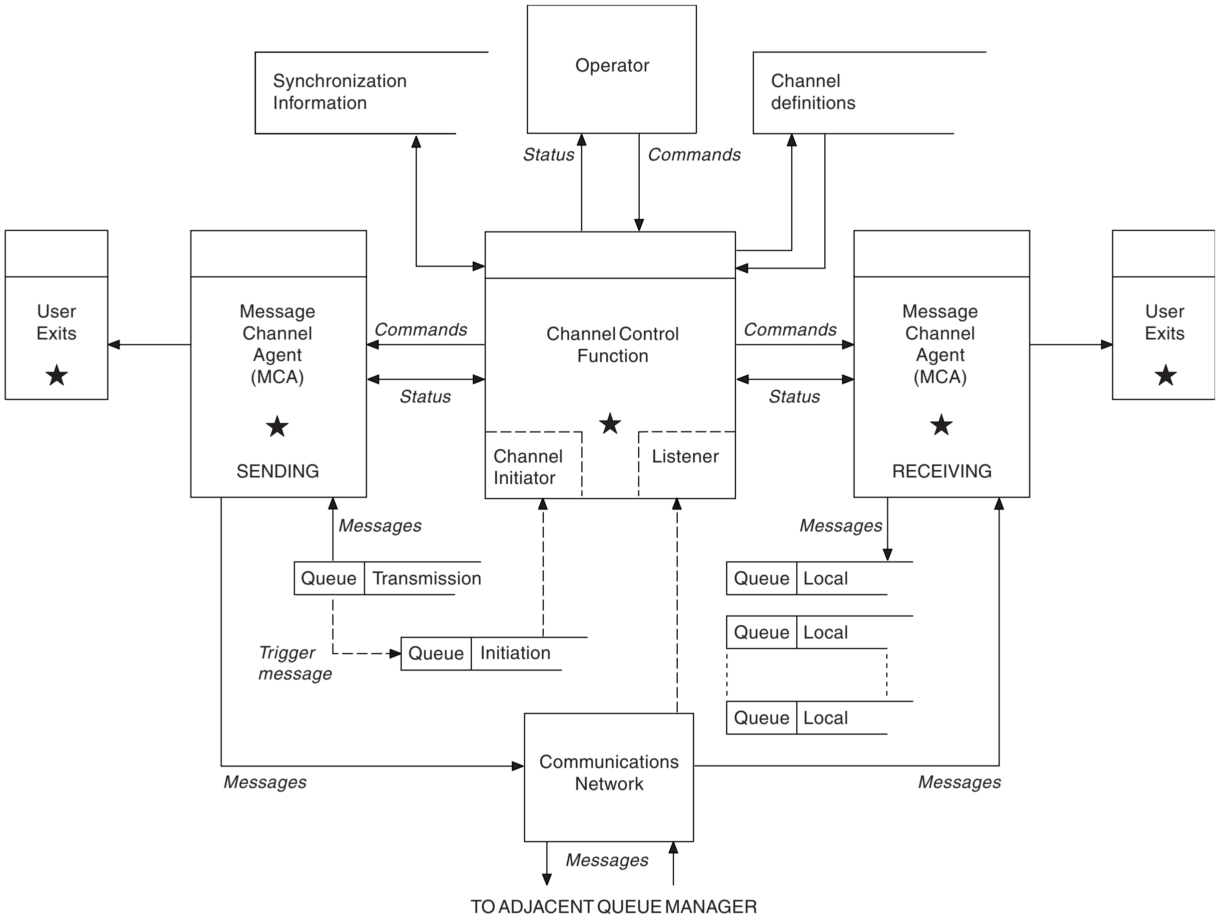


Figure 93. Distributed queue management model

Note:

1. There is one MCA per channel, depending on the platform. There might be one or more channel control functions for a particular queue manager.
2. The implementation of MCAs and channel control functions is highly platform-dependent. They can be programs or processes or threads, and they can be a single entity or many comprising several independent or linked parts.
3. All components marked with a star can use the MQI.

Channel parameters

An MCA receives its parameters in one of several ways:

- If started by a command, the channel name is passed in a data area. The MCA then reads the channel definition directly to obtain its attributes.
- For sender, and in some cases server channels, the MCA can be started automatically by the queue manager trigger. The channel name is retrieved from the trigger process definition, where applicable, and is passed to the MCA. The remaining processing is the same as previously described. Server channels must only be set up to trigger if they are fully qualified, that is, they specify a CONNAME to connect to.
- If started remotely by a sender, server, requester, or client-connection, the channel name is passed in the initial data from the partner message channel agent. The MCA reads the channel definition directly to obtain its attributes.

Certain attributes not defined in the channel definition are also negotiable:

Split messages
If one end does not support split messages then the split messages are not sent.

Conversion capability
If one end cannot perform the necessary code page conversion or numeric encoding conversion when needed, the other end must handle it. If neither end supports it, when needed, the channel cannot start.

Distribution list support
If one end does not support distribution lists, the partner MCA sets a flag in its transmission queue so that it knows to intercept messages intended for multiple destinations.

Channel status and sequence numbers

Message channel agent programs keep records of the current sequence number and logical unit of work number for each channel, and of the general status of the channel. Some platforms allow you to display this status information to help you control channels.

How to send a message to another queue manager

This section describes the simplest way to send a message between queue managers, including prerequisites and authorizations required. Other methods can also be used to send messages to a remote queue manager.

Before you send a message from one queue manager to another, you need to do the following steps:

1. Check that your chosen communication protocol is available.
2. Start the queue managers.
3. Start the channel initiators.
4. Start the listeners.

You also need to have the correct IBM MQ security authorization to create the objects required.

To send messages from one queue manager to another:

- Define the following objects on the source queue manager:
  - Sender channel
  - Remote queue definition
  - Initiation queue (z/OS required on z/OS, otherwise optional)
  - Transmission queue
  - Dead-letter queue
- Define the following objects on the target queue manager:
  - Receiver channel
  - Target queue
  - Dead-letter queue

You can use several different methods to define these objects, depending on your IBM MQ platform:

- On all platforms, you can use the IBM MQ script commands (MQSC) described in The MQSC commands the programmable command format (PCF) commands described in Automating administration tasks, or the IBM MQ Explorer.
- z/OS On z/OS, you can also use the Operation and Control panels described in Administering IBM MQ for z/OS .
- IBM i On IBM i, you can also use the panel interface.

See the following subtopics for more information on creating the components for sending messages to another queue manager:

Related concepts:

"IBM MQ distributed queuing techniques" on page 961
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 982
Distributed queue management (DQM) is used to define and control communication between queue managers.

"Triggering channels" on page 1008
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 1005
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.

z/OS "Setting up communications with other queue managers" on page 1531
This section describes the IBM MQ for z/OS preparations you need to make before you can start to use distributed queuing.

Related tasks:

Multi "Creating and managing queue managers on Multiplatforms" on page 833
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.

ULW "Monitoring and controlling channels on UNIX, Linux, and Windows" on page 1015
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.

IBM i "Monitoring and controlling channels on IBM i" on page 1039
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 server and client" on page 844
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 1063
Clusters provide a mechanism for interconnecting queue managers in a way that simplifies both the initial configuration and the ongoing management. You can define cluster components, and create and manage clusters.

Defining the channels:

To send messages from one queue manager to another, you must define two channels. You must define one channel on the source queue manager and one channel on the target queue manager.

On the source queue manager
Define a channel with a channel type of SENDER. You need to specify the following:

- The name of the transmission queue to be used (the XMITQ attribute).

- The connection name of the partner system (the CONNAME attribute).
- The name of the communication protocol you are using (the TRPTYPE attribute). On IBM MQ for z/OS, the protocol must be TCP or LU6.2. On other platforms, you do not have to specify this. You can leave it to pick up the value from your default channel definition.

Details of all the channel attributes are given in Channel attributes.

On the target queue manager

Define a channel with a channel type of RECEIVER, and the same name as the sender channel.

Specify the name of the communication protocol you are using (the TRPTYPE attribute). On IBM MQ for z/OS, the protocol must be TCP or LU6.2. On other platforms, you do not have to specify this. You can leave it to pick up the value from your default channel definition.

Receiver channel definitions can be generic. This means that if you have several queue managers communicating with the same receiver, the sending channels can all specify the same name for the receiver, and one receiver definition applies to them all.

When you have defined the channel, you can test it using the PING CHANNEL command. This command sends a special message from the sender channel to the receiver channel and checks that it is returned.

Defining the queues:

To send messages from one queue manager to another, you must define up to six queues. You must define up to four queues on the source queue manager, and up to two queues on the target queue manager.

On the source queue manager

- Remote queue definition

  In this definition, specify the following:

  Remote queue manager name
  The name of the target queue manager.

  Remote queue name
  The name of the target queue on the target queue manager.

  Transmission queue name
  The name of the transmission queue. You do not have to specify this transmission queue name. If you do not, a transmission queue with the same name as the target queue manager is used. If this does not exist, the default transmission queue is used. You are advised to give the transmission queue the same name as the target queue manager so that the queue is found by default.
- Initiation queue definition

  z/OS This is required. You must use the initiation queue called SYSTEM.CHANNEL.INITQ.

  Multi This is optional. Consider naming the initiation queue SYSTEM.CHANNEL.INITQ.
- Transmission queue definition

  A local queue with the USAGE attribute set to XMITQ. IBM i If you are using the IBM MQ for IBM i native interface, the USAGE attribute is *TMQ.
- Dead-letter queue definition

  Define a dead-letter queue to which undelivered messages can be written.

On the target queue manager

- Local queue definition

The target queue. The name of this queue must be the same as that specified in the remote queue name field of the remote queue definition on the source queue manager.

- Dead-letter queue definition

  Define a dead-letter queue to which undelivered messages can be written.

Related concepts:

"Creating a transmission queue"
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.

IBM i "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.

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.

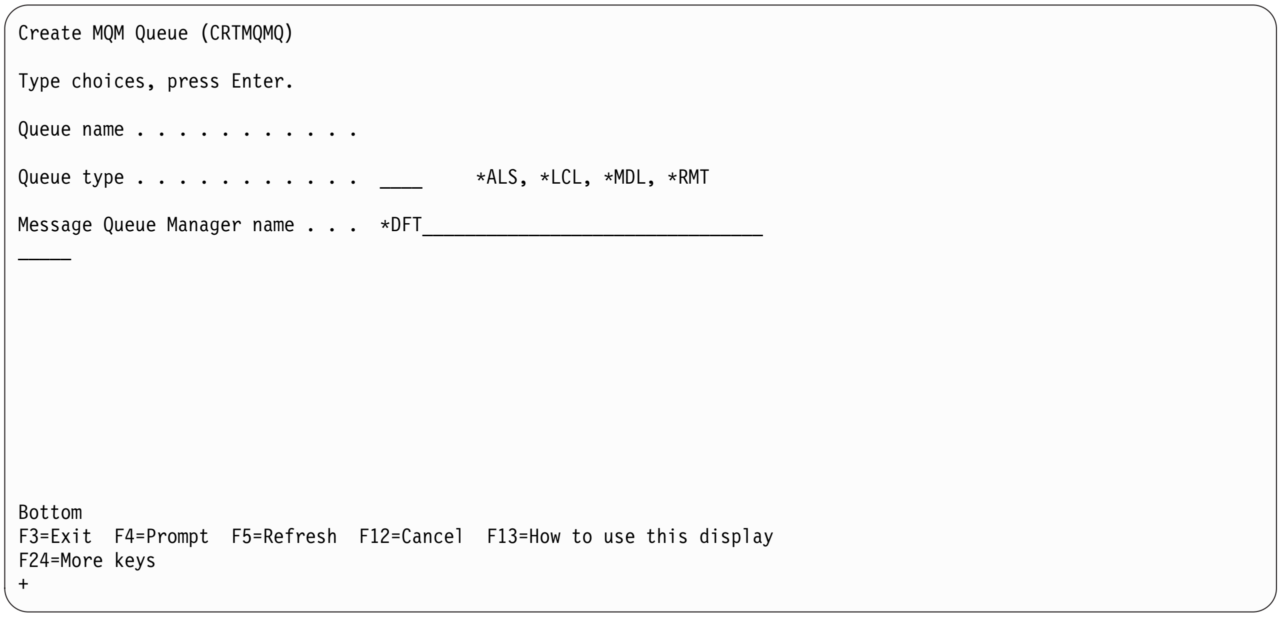
Creating a transmission queue on IBM i:

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 94 on page 989.

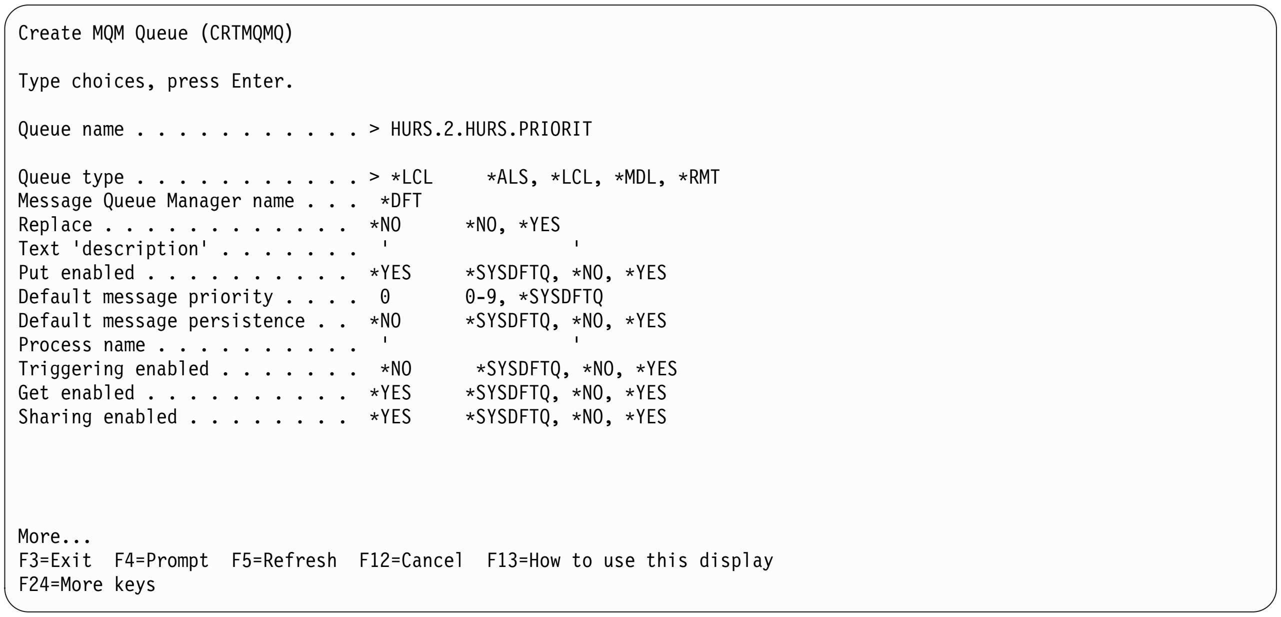


```
Create MQM Queue (CRTMQMQ)

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 94. 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.

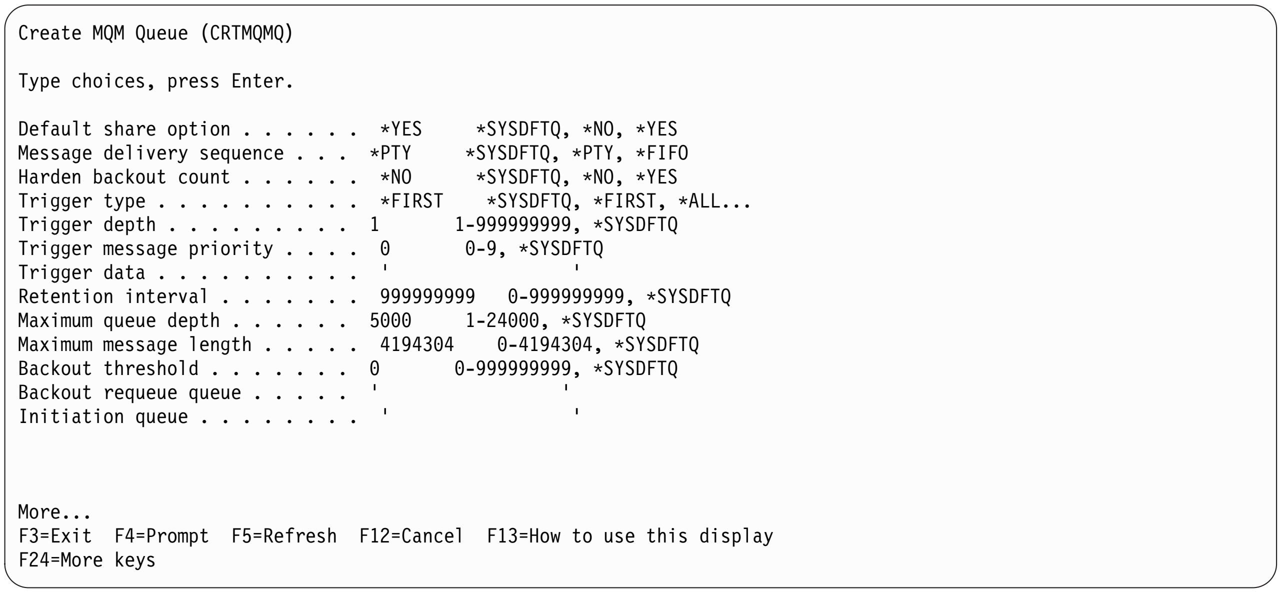
You are presented with the second page of the Create MQM Queue panel; see Figure 95.



```
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 95. Create a queue (2)

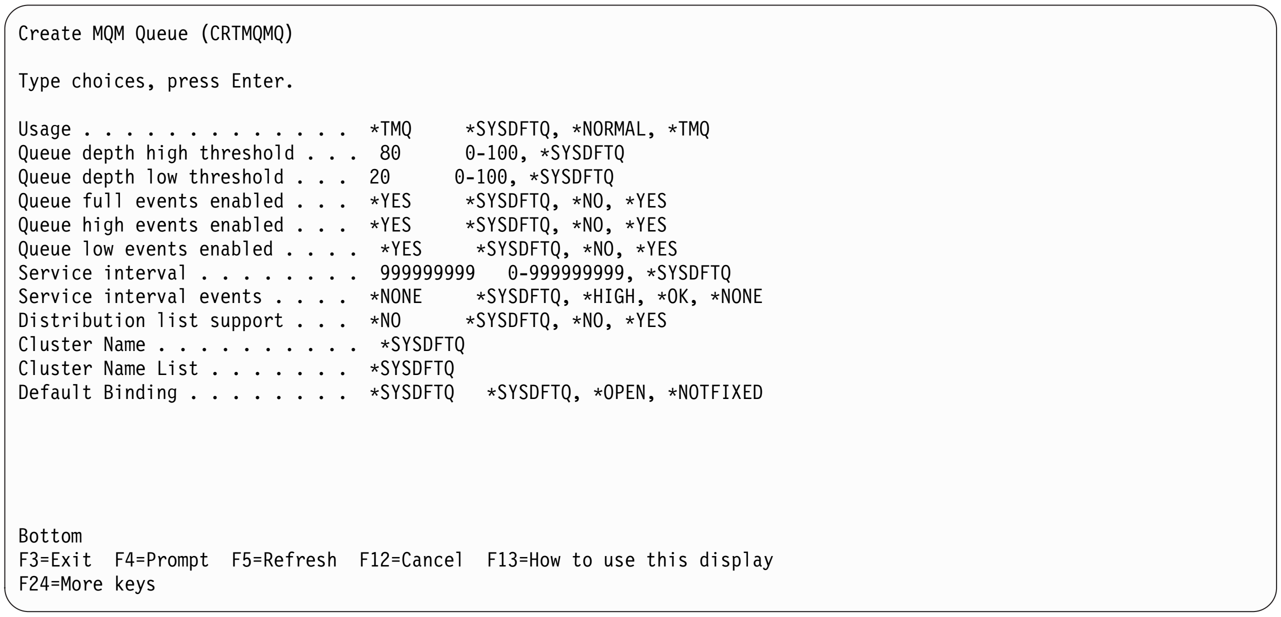
Change any of the default values shown. Press page down to scroll to the next screen; see Figure 96 on page 990.



```
Create MQM Queue (CRTMQMQ)

Type choices, press Enter.

Default share option . . . . . .  *YES     *SYSDFTQ, *NO, *YES
Message delivery sequence . . .  *PTY     *SYSDFTQ, *PTY, *FIFO
Harden backout count . . . . . .  *NO      *SYSDFTQ, *NO, *YES
Trigger type . . . . . . . . . .  *FIRST    *SYSDFTQ, *FIRST, *ALL...
Trigger depth . . . . . . . . .  1       1-999999999, *SYSDFTQ
Trigger message priority . . . .  0       0-9, *SYSDFTQ
Trigger data . . . . . . . . . .  '                 '
Retention interval . . . . . . .  999999999   0-999999999, *SYSDFTQ
Maximum queue depth . . . . . .  5000     1-24000, *SYSDFTQ
Maximum message length . . . . .  4194304    0-4194304, *SYSDFTQ
Backout threshold . . . . . . .  0       0-999999999, *SYSDFTQ
Backout requeue queue . . . . .  '                 '
Initiation queue . . . . . . . .  '                 '



More...
F3=Exit  F4=Prompt  F5=Refresh  F12=Cancel  F13=How to use this display
F24=More keys
```

Figure 96. 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 97. 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 UNIX, Linux, and Windows” on page 1015.

The commands fall into the following groups:
- Channel administration
- Channel control
- Channel status monitoring

Channel administration commands deal with the definitions of the channels. They enable you to:
- Create a channel definition
- Copy a channel definition
- Alter a channel definition
- Delete a channel definition

Channel control commands manage the operation of the channels. They enable you to:
- Start a channel
- Stop a channel
- Re-synchronize with partner (in some implementations)
- Reset message sequence numbers
- Resolve an in-doubt batch of messages
- Ping; send a test communication across the channel

Channel monitoring displays the state of channels, for example:
- Current channel settings

- Whether the channel is active or inactive
- Whether the channel terminated in a synchronized state

For more information about defining, controlling and monitoring channels, see the following subtopics:

Preparing channels:

Before trying to start a message channel or MQI channel, you must prepare the channel. You must make sure that all the attributes of the local and remote channel definitions are correct and compatible.

Channel attributes describes the channel definitions and attributes.

Although you set up explicit channel definitions, the channel negotiations carried out when a channel starts, might override one or other of the values defined. This behavior is normal, and not apparent to the user, and has been arranged in this way so that otherwise incompatible definitions can work together.

Auto-definition of receiver and server-connection channels

In IBM MQ on all platforms except z/OS, if there is no appropriate channel definition, then for a receiver or server-connection channel that has auto-definition enabled, a definition is created automatically. The definition is created using:

1. The appropriate model channel definition, SYSTEM.AUTO.RECEIVER, or SYSTEM.AUTO.SVRCONN. The model channel definitions for auto-definition are the same as the system defaults, SYSTEM.DEF.RECEIVER, and SYSTEM.DEF.SVRCONN, except for the description field, which is "Auto-defined by" followed by 49 blanks. The systems administrator can choose to change any part of the supplied model channel definitions.
2. Information from the partner system. The values from the partner are used for the channel name and the sequence number wrap value.
3. A channel exit program, which you can use to alter the values created by the auto-definition. See Channel auto-definition exit program.

The description is then checked to determine whether it has been altered by an auto-definition exit or because the model definition has been changed. If the first 44 characters are still "Auto-defined by" followed by 29 blanks, the queue manager name is added. If the final 20 characters are still all blanks the local time and date are added.

When the definition has been created and stored the channel start proceeds as though the definition had always existed. The batch size, transmission size, and message size are negotiated with the partner.

Defining other objects

Before a message channel can be started, both ends must be defined (or enabled for auto-definition) at their queue managers. The transmission queue it is to serve must be defined to the queue manager at the sending end. The communication link must be defined and available. It might be necessary for you to prepare other IBM MQ objects, such as remote queue definitions, queue manager alias definitions, and reply-to queue alias definitions, to implement the scenarios described in “Configuring distributed queuing” on page 961.

For information about defining MQI channels, see “Defining MQI channels” on page 858.

Multiple message channels per transmission queue

It is possible to define more than one channel per transmission queue, but only one of these channels can be active at any one time. Consider this option for the provision of alternative routes between queue managers for traffic balancing and link failure corrective action. A transmission queue cannot be used by another channel if the previous channel to use it terminated leaving a batch of messages in-doubt at the sending end. For more information, see "In-doubt channels" on page 1004.

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 98 on page 994 shows the hierarchy of all possible channel states and the substates that apply to each of the channel states.

Figure 99 on page 995 shows the links between channel states. These links apply to all types of message channel and server-connection channels.

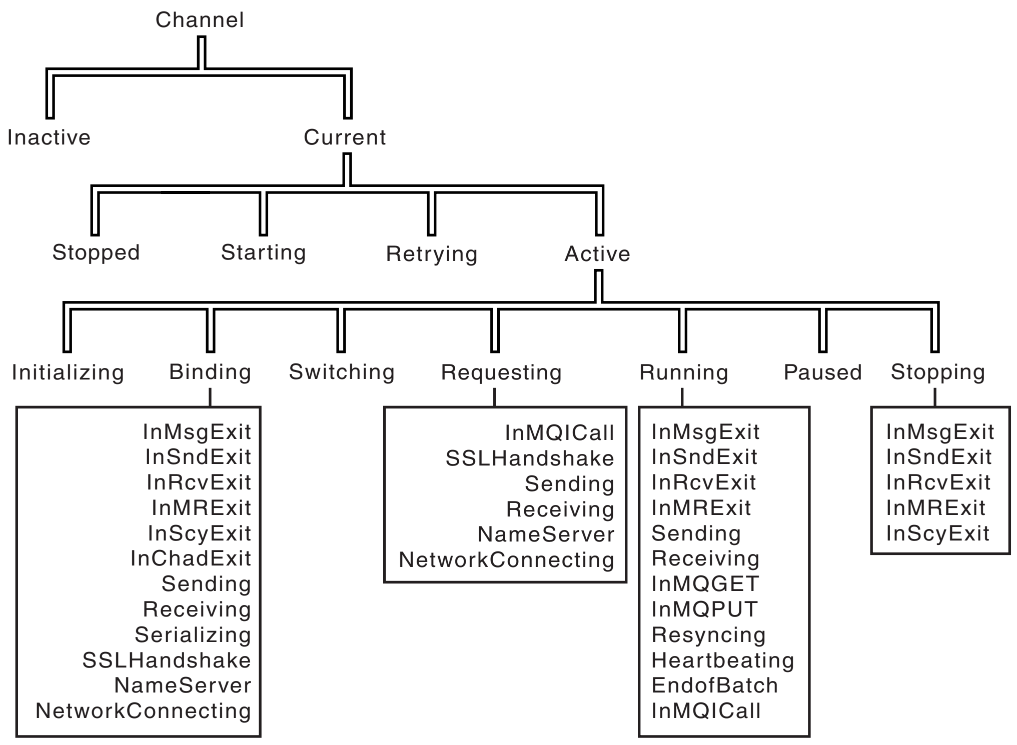


Figure 98. Channel states and substates

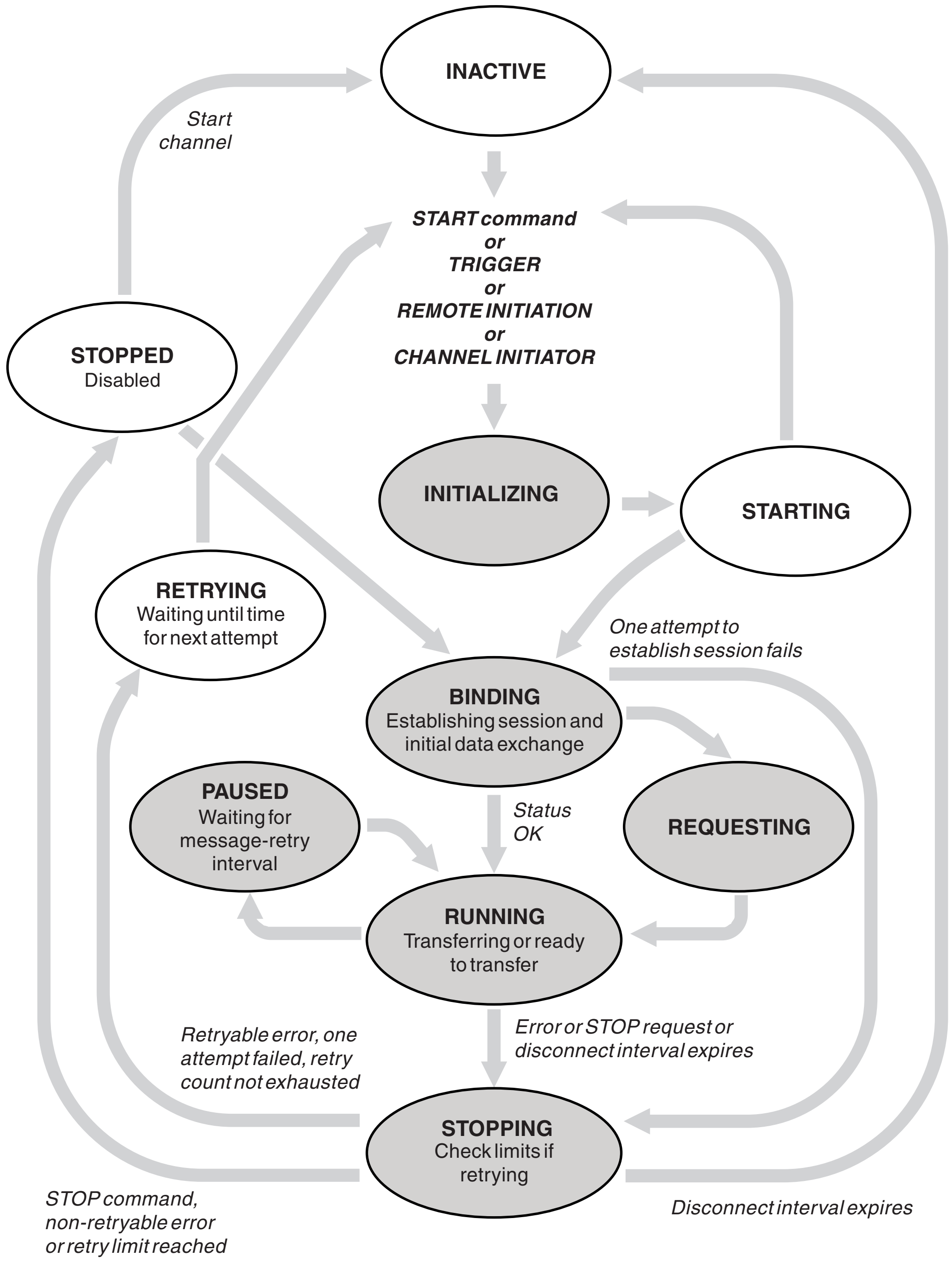

INACTIVE
Start channel
START command or TRIGGER or REMOTE INITIATION or CHANNEL INITIATOR
STOPPED Disabled
INITIALIZING
STARTING
RETRYING Waiting until time for next attempt
One attempt to establish session fails
BINDING Establishing session and initial data exchange
PAUSED Waiting for message-retry interval
Status OK
REQUESTING
RUNNING Transferring or ready to transfer
Retryable error, one attempt failed, retry count not exhausted
Error or STOP request or disconnect interval expires
STOPPING Check limits if retrying
STOP command, non-retryable error or retry limit reached
Disconnect interval expires

Figure 99. 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 99.

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 98 on page 994.

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.

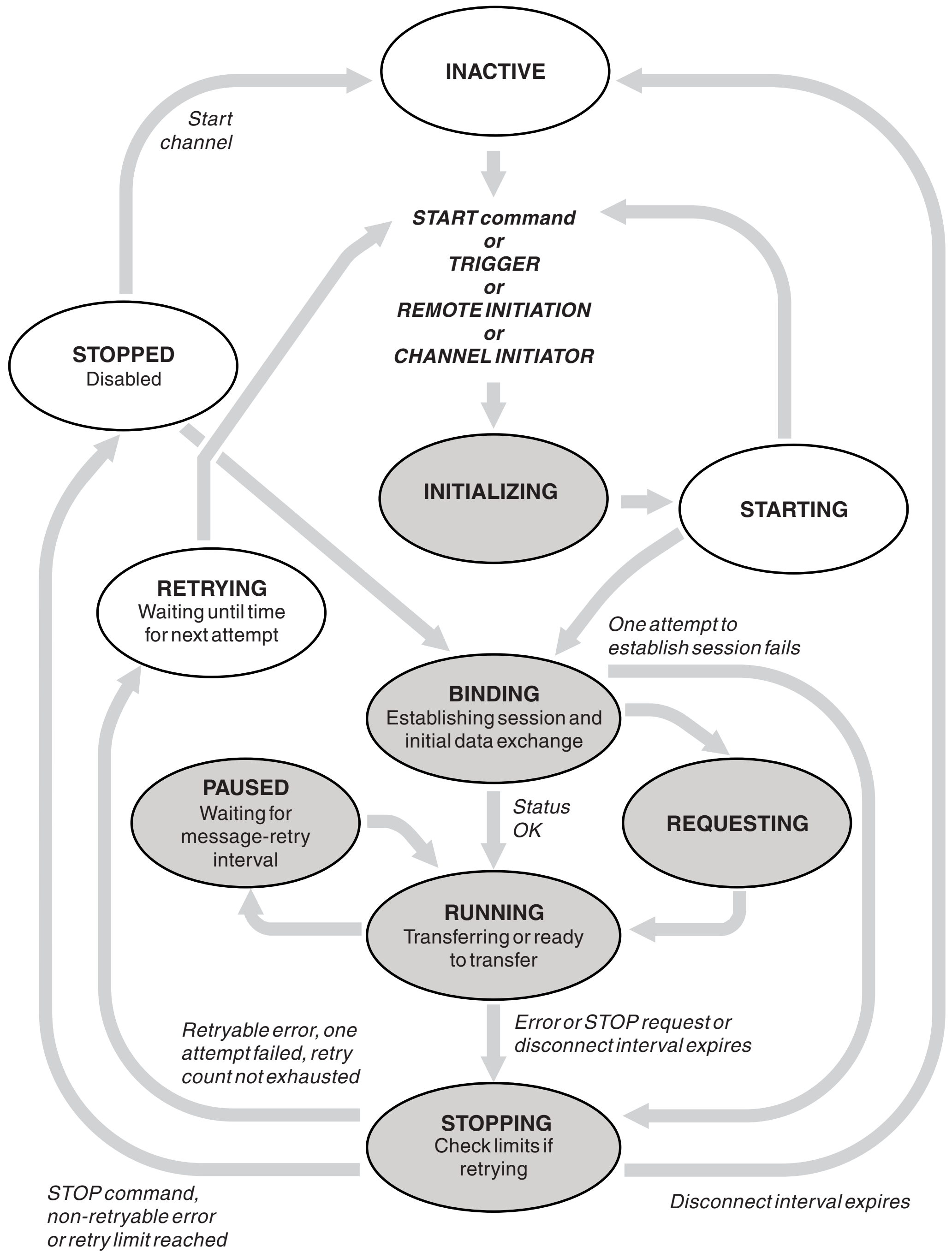

INACTIVE
Start channel
START command or TRIGGER or REMOTE INITIATION or CHANNEL INITIATOR
STOPPED Disabled
INITIALIZING
STARTING
RETRYING Waiting until time for next attempt
One attempt to establish session fails
BINDING Establishing session and initial data exchange
PAUSED Waiting for message-retry interval
Status OK
REQUESTING
RUNNING Transferring or ready to transfer
Retryable error, one attempt failed, retry count not exhausted
Error or STOP request or disconnect interval expires
STOPPING Check limits if retrying
STOP command, non-retryable error or retry limit reached
Disconnect interval expires

Figure 100. Flows between channel states

Note:

1. When a channel is in one of the six states highlighted in Figure 100 on page 996 (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:

- z/OS Use the `ALTER QMGR MAXCHL` command.
- IBM i Edit the queue manager initialization file.
- UNIX Linux 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:

- ULW Administering IBM MQ.
- IBM i Administering IBM MQ for IBM i.
- z/OS 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 z/OS or the z/OS console, and an event is generated. When a slot is later freed and the channel is able to acquire it, another message and event are generated. Neither of these events and messages are generated if the channel is able to acquire a slot straight away.

If a STOP CHANNEL command is issued while the channel is waiting to become active, the channel goes to STOPPED state. A Channel-Stopped event is raised.

Server-connection channels are included in the maximum number of active channels.

For more information about specifying the maximum number of active channels, see the following topics:

- ULW Administering IBM MQ.
- IBM i Administering IBM MQ for IBM i.
- z/OS 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 1003).

Note: For IBM i IBM i, UNIX, Linux, and Windows systems, a channel initiator must be running for retry to be attempted. If the channel initiator is not available, the channel becomes inactive and must be manually restarted. If you are using a script to start the channel, ensure that the channel initiator is running before you try to run the script.

Long retry count (LONGRTY) describes how retrying works. If the error clears, the channel restarts automatically, and the transmission queue is re-enabled. If the retry limit is reached without the error clearing, the channel goes to STOPPED state. A stopped channel must be restarted manually by the operator. If the error is still present, it does not retry again. When it does start successfully, the transmission queue is re-enabled.

z/OS If the channel initiator stops while a channel is in RETRYING or STOPPED status, the channel status is remembered when the channel initiator is restarted. However, the channel status for the SVRCONN channel type is reset if the channel initiator stops while the channel is in STOPPED status.

Multi 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. However, the channel status for the SVRCONN channel type is reset if the queue manager stops while the channel is in STOPPED status.

If a channel is unable to put a message to the target queue because that queue is full or put inhibited, the channel can retry the operation a number of times (specified in the message-retry count attribute) at a time interval (specified in the message-retry interval attribute). Alternatively, you can write your own message-retry exit that determines which circumstances cause a retry, and the number of attempts made. The channel goes to PAUSED state while waiting for the message-retry interval to finish.

See Channel attributes for information about the channel attributes, and Channel-exit programs for messaging channels for information about the message-retry exit.

Server-connection channel limits:

You can set server-connection channel limits to prevent client applications from exhausting queue manager channel resources, **MAXINST**, and to prevent a single client application from exhausting server-connection channel capacity, **MAXINSTC**.

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.

Checking that the other end of the channel is still available:

You can use the heartbeat interval, the keep alive interval, and the receive timeout, to check that the other end of the channel is available.

Heartbeats

You can use the heartbeat interval channel attribute to specify that flows are to be passed from the sending MCA when there are no messages on the transmission queue, as is described in Heartbeat interval (HBINT).

Keep alive

In IBM MQ for z/OS, if you are using TCP/IP as the transport protocol, you can also specify a value for the **Keepalive** interval channel attribute (KAINT). You are recommended to give the **Keepalive** interval a higher value than the heartbeat interval, and a smaller value than the disconnect value. You can use this attribute to specify a time-out value for each channel, as is described in Keepalive Interval (KAINT).

In IBM MQ for IBM i, UNIX, Linux, and Windows systems, if you are using TCP as your transport protocol, you can set `keepalive=yes`. If you specify this option, TCP periodically checks that the other end of the connection is still available. It is not, the channel is terminated. This option is described in Keepalive Interval (KAINT).

If you have unreliable channels that report TCP errors, use of the **Keepalive** option means that your channels are more likely to recover.

You can specify time intervals to control the behavior of the **Keepalive** option. When you change the time interval, only TCP/IP channels started after the change are affected. Ensure that the value that you choose for the time interval is less than the value of the disconnect interval for the channel.

For more information about using the **Keepalive** option, see the KAINT parameter in the DEFINE CHANNEL command.

Receive timeout

If you are using TCP as your transport protocol, the receiving end of an idle non-MQI channel connection is also closed if no data is received for a period. This period, the *receive time-out* value, is determined according to the HBINT (heartbeat interval) value.

In IBM MQ for IBM i, UNIX, Linux, and Windows systems, the *receive time-out* value is set as follows:
1. For an initial number of flows, before any negotiation takes place, the *receive time-out* value is twice the HBINT value from the channel definition.
2. After the channels negotiate an HBINT value, if HBINT is set to less than 60 seconds, the *receive time-out* value is set to twice this value. If HBINT is set to 60 seconds or more, the *receive time-out* value is set to 60 seconds greater than the value of HBINT.

In IBM MQ for z/OS, the *receive time-out* value is set as follows:
1. For an initial number of flows, before any negotiation takes place, the *receive time-out* value is twice the HBINT value from the channel definition.
2. If RCVTIME is set, the timeout is set to one of
   - the negotiated HBINT multiplied by a constant
   - the negotiated HBINT plus a constant number of seconds
   - a constant number of seconds

depending on the RCVTTYPE parameter, and subject to any limit imposed by RCVTMIN if it applies. RCVTMIN does not apply when RCVTTYPE(EQUAL) is configured. If you use a constant value of RCVTIME and you use a heartbeat interval, do not specify an RCVTIME less than the heartbeat interval. For details of the RCVTIME, RCVTMIN and RCVTTYPE attributes, see the ALTER QMGR command.

Note:

1. If either of the values is zero, there is no timeout.
2. For connections that do not support heartbeats, the HBINT value is negotiated to zero in step 2 and hence there is no timeout, so you must use TCP/IP KEEPALIVE.
3. For client connections that use sharing conversations, heartbeats can flow across the channel (from both ends) all the time, not just when an MQGET is outstanding.
4. For client connections where sharing conversations are not in use, heartbeats are flowed from the server only when the client issues an MQGET call with wait. Therefore, you are not recommended to set the heartbeat interval too small for client channels. For example, if the heartbeat is set to 10 seconds, an MQCMIT call fails (with MQRC_CONNECTION_BROKEN) if it takes longer than 20 seconds to commit because no data flowed during this time. This can happen with large units of work. However, it does not happen if appropriate values are chosen for the heartbeat interval because only MQGET with wait takes significant periods of time.

   Provided SHARECNV is not zero, the client uses a full duplex connection, which means that the client can (and does) heartbeat during all MQI calls
5. In IBM MQ Version 7 Client channels, heartbeats can flow from both the server as well as the client side. The timeout at either end is based upon 2*HBINT for HBINTs of less than 60 seconds and HBINT+60 for HBINTs of over 60 seconds.
6. Canceling the connection after twice the heartbeat interval is valid because a data or heartbeat flow is expected at least at every heartbeat interval. Setting the heartbeat interval too small, however, can cause problems, especially if you are using channel exits. For example, if the HBINT value is one second, and a send or receive exit is used, the receiving end waits for only 2 seconds before canceling the channel. If the MCA is performing a task such as encrypting the message, this value might be too short.

Adopting an MCA:

The Adopt MCA function enables IBM MQ to cancel a receiver channel and start a new one in its place.

If a channel loses contact, the receiver channel can be left in a 'communications receive' state. When communications are re-established the sender channel attempts to reconnect. If the remote queue manager finds that the receiver channel is already running it does not allow another version of the same receiver channel to start. This problem requires user intervention to rectify the problem or the use of system keepalive.

The Adopt MCA function solves the problem automatically. It enables IBM MQ to cancel a receiver channel and to start a new one in its place.

Related information:
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
- z/OS IBM i other platform-specific mechanisms, as follows:

  z/OS **For z/OS:**
  The Stop a channel panel

  IBM i **For IBM i:**
  The ENDMQMCHL CL command or the END option on the WRKMQMCHL panel

There are three options for stopping channels using these commands:

QUIESCE
The QUIESCE option attempts to end the current batch of messages before stopping the channel.

FORCE
The FORCE option attempts to stop the channel immediately and might require the channel to resynchronize when it restarts because the channel might be left in doubt.

z/OS On IBM MQ for z/OS, FORCE interrupts any message reallocation in progress, which might leave BIND_NOT_FIXED messages partially reallocated or out of order.

TERMINATE
The TERMINATE option attempts to stop the channel immediately, and terminates the thread or process of the channel.

z/OS On IBM MQ for z/OS, TERMINATE interrupts any message reallocation in progress, which might leave BIND_NOT_FIXED messages partially reallocated or out of order.

All these options leave the channel in a STOPPED state, requiring operator intervention to restart it.

Stopping the channel at the sending end is effective but does require operator intervention to restart. At the receiving end of the channel, things are much more difficult because the MCA is waiting for data from the sending side, and there is no way to initiate an *orderly* termination of the channel from the receiving side; the stop command is pending until the MCA returns from its wait for data.

Consequently there are three recommended ways of using channels, depending upon the operational characteristics required:

- If you want your channels to be long running, note that there can be orderly termination only from the sending end. When channels are interrupted, that is, stopped, operator intervention (a START CHANNEL command) is required in order to restart them.
- If you want your channels to be active only when there are messages for them to transmit, set the disconnect interval to a fairly low value. The default setting is high and so is not recommended for channels where this level of control is required. Because it is difficult to interrupt the receiving channel, the most economical option is to have the channel automatically disconnect and reconnect as the workload demands. For most channels, the appropriate setting of the disconnect interval can be established heuristically.
- You can use the heartbeat-interval attribute to cause the sending MCA to send a heartbeat flow to the receiving MCA during periods in which it has no messages to send. This action releases the receiving MCA from its wait state and gives it an opportunity to quiesce the channel without waiting for the disconnect interval to expire. Give the heartbeat interval a lower value than the value of the disconnect interval.

  Note:

  1. You are advised to set the disconnect interval to a low value, or to use heartbeats, for server channels. This low value is to allow for the case where the requester channel ends abnormally (for example, because the channel was canceled) when there are no messages for the server channel to send. If the disconnect interval is set high and heartbeats are not in use, the server does not detect that the requester has ended (which it will only do the next time it tries to send a message to the requester). While the server is still running, it holds the transmission queue open for exclusive input in order to get any more messages that arrive on the queue. If an attempt is made to restart the channel from the requester, the start request receives an error because the server still has the transmission queue open for exclusive input. It is necessary to stop the server channel, and then restart the channel from the requester again.

Restarting stopped channels:

When a channel goes into STOPPED state, you have to restart the channel manually.

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.

z/OS If the channel initiator stops while a channel is in RETRYING or STOPPED status, the channel status is remembered when the channel initiator is restarted. However, the channel status for the SVRCONN channel type is reset if the channel initiator stops while the channel is in STOPPED status.

Multi If the queue manager stops while a channel is in RETRYING or STOPPED status, the channel status is remembered when the queue manager is restarted. From IBM MQ Version 8.0 onwards, this applies to SVRCONN channels as well. Previously, the channel status for the SVRCONN channel type was reset if the channel initiator stopped while the channel was in STOPPED status.

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

- z/OS On z/OS, by using the Start a channel panel.
- IBM i On IBM i, by using either the STRMQMCHL CL command or the START option on the WRKMQMCHL panel.

In-doubt channels:

An in-doubt channel is a channel that is in doubt with a remote channel about which messages have been sent and received.

Note the distinction between this and a queue manager being in doubt about which messages should be committed to a queue.

You can reduce the opportunity for a channel to be placed in doubt by using the Batch Heartbeat channel parameter (BATCHHB). When a value for this parameter is specified, a sender channel checks that the remote channel is still active before taking any further action. If no response is received the receiver channel is considered to be no longer active. The messages can be rolled-back, and re-routed, and the sender-channel is not put in doubt. This reduces the time when the channel could be placed in doubt to the period between the sender channel verifying that the receiver channel is still active, and verifying that the receiver channel has received the sent messages. See Channel attributes for more information about the batch heartbeat parameter.

In-doubt channel problems are typically resolved automatically. Even when communication is lost, and a channel is placed in doubt with a message batch at the sender with receipt status unknown, the situation is resolved when communication is re-established. Sequence number and LUWID records are kept for this purpose. The channel is in doubt until LUWID information has been exchanged, and only one batch of messages can be in doubt for the channel.

You can, when necessary, resynchronize the channel manually. The term *manual* includes use of operators or programs that contain IBM MQ system management commands. The manual resynchronization process works as follows. This description uses MQSC commands, but you can also use the PCF equivalents.

1. Use the DISPLAY CHSTATUS command to find the last-committed logical unit of work ID (LUWID) for ***each*** side of the channel. Do this using the following commands:
   - For the in-doubt side of the channel:
     ```
     DISPLAY CHSTATUS( name ) SAVED CURLUWID
     ```
     You can use the CONNAME and XMITQ parameters to further identify the channel.
   - For the receiving side of the channel:
     ```
     DISPLAY CHSTATUS( name ) SAVED LSTLUWID
     ```
     You can use the CONNAME parameter to further identify the channel.

   The commands are different because only the sending side of the channel can be in doubt. The receiving side is never in doubt.

   On IBM MQ for IBM i, the DISPLAY CHSTATUS command can be executed from a file using the STRMQMMQSC command or the Work with MQM Channel Status CL command, WRKMQMCHST
2. If the two LUWIDs are the same, the receiving side has committed the unit of work that the sender considers to be in doubt. The sending side can now remove the in-doubt messages from the transmission queue and re-enable it. This is done with the following channel RESOLVE command:
   ```
   RESOLVE CHANNEL( name ) ACTION(COMMIT)
   ```
3. If the two LUWIDs are different, the receiving side has not committed the unit of work that the sender considers to be in doubt. The sending side needs to retain the in-doubt messages on the transmission queue and re-send them. This is done with the following channel RESOLVE command:
   ```
   RESOLVE CHANNEL( name ) ACTION(BACKOUT)
   ```

IBM i On IBM MQ for IBM i, you can use the Resolve MQM Channel command, RSVMQMCHL.

Once this process is complete the channel is no longer in doubt. The transmission queue can now be used by another channel, if required.

Problem determination:

There are two distinct aspects to problem determination - problems discovered when a command is being submitted, and problems discovered during operation of the channels.

Command validation

Commands and panel data must be free from errors before they are accepted for processing. Any errors found by the validation are immediately notified to the user by error messages.

Problem diagnosis begins with the interpretation of these error messages and taking corrective action.

Processing problems

Problems found during normal operation of the channels are notified to the system console or the system log. Problem diagnosis begins with the collection of all relevant information from the log, and continues with analysis to identify the problem.

Confirmation and error messages are returned to the terminal that initiated the commands, when possible.

IBM MQ produces accounting and statistical data, which you can use to identify trends in utilization and performance. Multi On Multiplatforms, this information is produced as PCF records, see Structure data types. z/OS On z/OS, this information is produced as SMF records, see Monitoring performance and resource usage.

Messages and codes

For messages and codes to help with the primary diagnosis of the problem, see Diagnostic messages and reason codes.

Safety of messages

In addition to the typical recovery features of IBM MQ, distributed queue management ensures that messages are delivered properly by using a sync point procedure coordinated between the two ends of the message channel. If this procedure detects an error, it closes the channel so that you can investigate the problem, and keeps the messages safely in the transmission queue until the channel is restarted.

The sync point procedure has an added benefit in that it attempts to recover an *in-doubt* situation when the channel starts. (*In-doubt* is the status of a unit of recovery for which a sync point has been requested but the outcome of the request is not yet known.) Also associated with this facility are the two functions:

1. Resolve with commit or backout
2. Reset the sequence number

The use of these functions occurs only in exceptional circumstances because the channel recovers automatically in most cases.

Fast, nonpersistent messages

The nonpersistent message speed (NPMSPEED) channel attribute can be used to specify that any nonpersistent messages on the channel are to be delivered more quickly. For more information about this attribute, see Nonpersistent message speed (NPMSPEED).

If a channel terminates while fast, nonpersistent messages are in transit, the messages might be lost and it is up to the application to arrange for their recovery if required.

If the receiving channel cannot put the message to its destination queue then it is placed on the dead letter queue, if one has been defined. If not, the message is discarded.

Note: If the other end of the channel does not support the option, the channel runs at normal speed.

Undelivered Messages

For information about what happens when a message cannot be delivered, see "What happens when a message cannot be delivered?."

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 101 shows the processing that occurs when an MCA is unable to put a message to the destination queue. (The options shown do not apply on all platforms.)

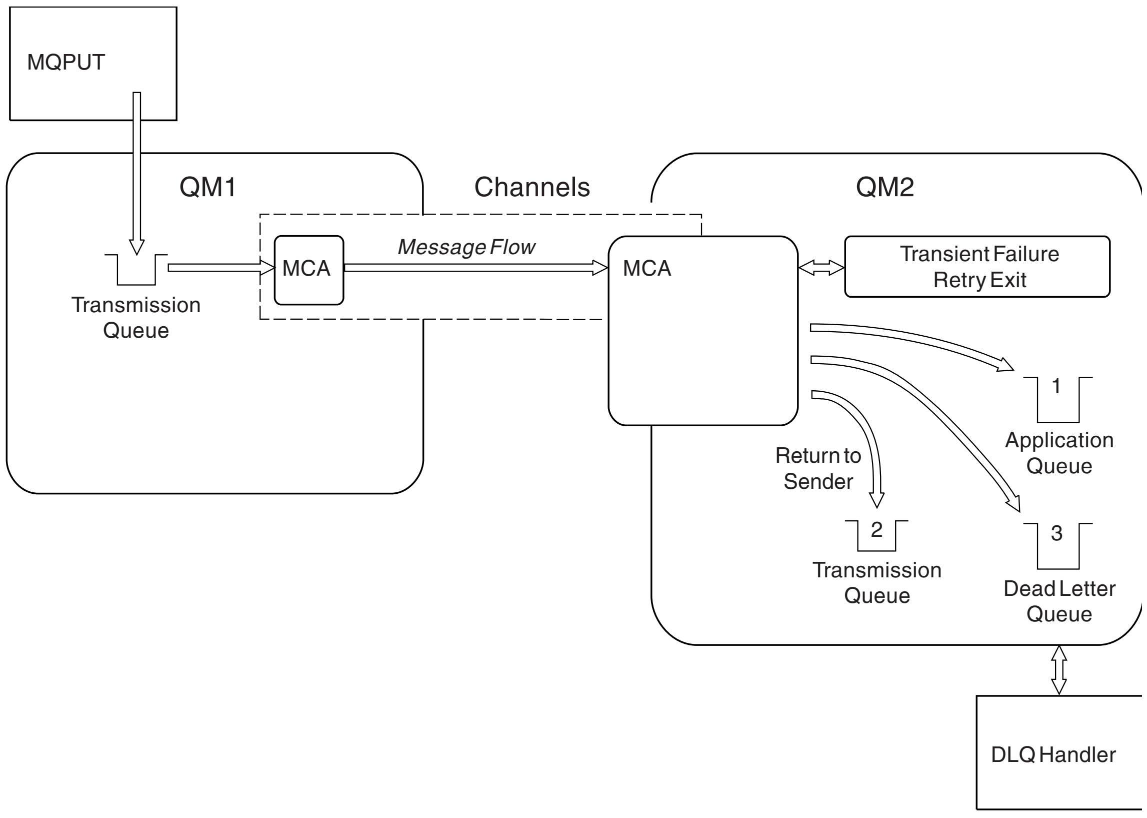


Figure 101. What happens when a message cannot be delivered

As shown in the figure, the MCA can do several things with a message that it cannot deliver. The action taken is determined by options specified when the channel is defined and on the MQPUT report options for the message.

1. Message-retry

   If the MCA is unable to put a message to the target queue for a reason that could be transitory (for example, because the queue is full), the MCA can wait and try the operation again later. You can determine if the MCA waits, for how long, and how many times it tries.

   - You can specify a message-retry time and interval for MQPUT errors when you define your channel. If the message cannot be put to the destination queue because the queue is full, or is inhibited for puts, the MCA tries the operation the number of times specified, at the time interval specified.
   - You can write your own message-retry exit. The exit enables you to specify under what conditions you want the MCA to try the MQPUT or MQOPEN operation again. Specify the name of the exit when you define the channel.

2. Return-to-sender

   If message-retry was unsuccessful, or a different type of error was encountered, the MCA can send the message back to the originator. To enable return-to-sender, you need to specify the following options in the message descriptor when you put the message to the original queue:

   - The MQRO_EXCEPTION_WITH_FULL_DATA report option
   - The MQRO_DISCARD_MSG report option
   - The name of the reply-to queue and reply-to queue manager

   If the MCA is unable to put the message to the destination queue, it generates an exception report containing the original message, and puts it on a transmission queue to be sent to the reply-to queue specified in the original message. (If the reply-to queue is on the same queue manager as the MCA, the message is put directly to that queue, not to a transmission queue.)

3. Dead-letter queue

   If a message cannot be delivered or returned, it is put on to the dead-letter queue (DLQ). You can use the DLQ handler to process the message. This processing is described in Processing messages on a dead-letter queue for IBM MQ for UNIX, Linux and Windows systems, and in The dead-letter queue handler utility (CSQUDLQH) for z/OS systems. If the dead-letter queue is not available, the sending MCA leaves the message on the transmission queue, and the channel stops. On a fast channel, nonpersistent messages that cannot be written to a dead-letter queue are lost.

   On IBM WebSphere MQ Version 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 information:

Use Dead-Letter Queue (USEDLQ)

Triggering channels

IBM MQ provides a facility for starting an application automatically when certain conditions on a queue are met. This facility is called triggering.

This explanation is intended as an overview of triggering concepts. For a complete description, see Starting IBM MQ applications using triggers.

For platform-specific information see the following:

- For Windows, see UNIX and Linux systems, "Triggering channels on UNIX, Linux, and Windows." on page 1009
- IBM i For IBM i, see "Triggering channels in IBM MQ for IBM i" on page 1010
- z/OS For z/OS, see "Transmission queues and triggering channels" on page 1533

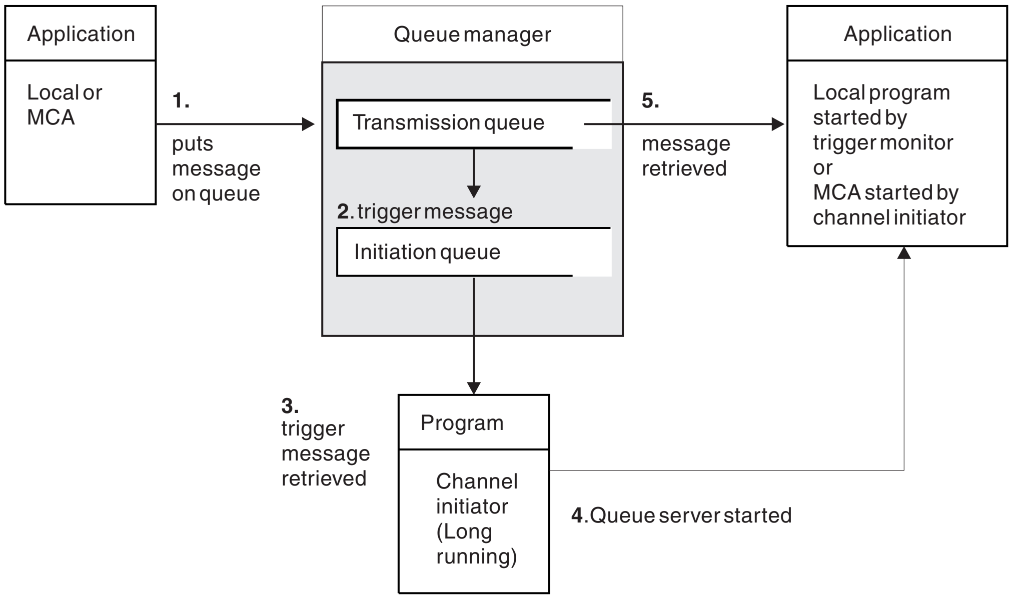


Figure 102. The concepts of triggering

The objects required for triggering are shown in Figure 102. It shows the following sequence of events:

1. The local queue manager places a message from an application or from a message channel agent (MCA) on the transmission queue.
2. When the triggering conditions are fulfilled, the local queue manager places a trigger message on the initiation queue.
3. The long-running channel initiator program monitors the initiation queue, and retrieves messages as they arrive.
4. The channel initiator processes the trigger messages according to information contained in them. This information might include the channel name, in which case the corresponding MCA is started.
5. The local application or the MCA, having been triggered, retrieves the messages from the transmission queue.

To set up this scenario, you need to:

- Create the transmission queue with the name of the initiation queue (that is, SYSTEM.CHANNEL.INITQ) in the corresponding attribute.
- Ensure that the initiation queue (SYSTEM.CHANNEL.INITQ) exists.
- Ensure that the channel initiator program is available and running. The channel initiator program must be provided with the name of the initiation queue in its start command. z/OS On z/OS, the name of the initiation queue is fixed, so is not used on the start command.
- Optionally, create the process definition for the triggering, if it does not exist, and ensure that the *UserData* field contains the name of the channel it serves. Instead of creating a process definition, you can specify the channel name in the **TriggerData** attribute of the transmission queue. IBM MQ for IBM i IBM i, UNIX, Linux, and Windows systems, allow the channel name to be specified as blank, in which case the first available channel definition with this transmission queue is used.
- Ensure that the transmission queue definition contains the name of the process definition to serve it, (if applicable), the initiation queue name, and the triggering characteristics you feel are most suitable. The trigger control attribute allows triggering to be enabled, or not, as necessary.

Note:

1. The channel initiator program acts as a 'trigger monitor' monitoring the initiation queue used to start channels.
2. An initiation queue and trigger process can be used to trigger any number of channels.
3. Any number of initiation queues and trigger processes can be defined.
4. A trigger type of FIRST is recommended, to avoid flooding the system with channel starts.

ULW

Triggering channels on UNIX, Linux, and Windows.

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.

IBM i

Triggering channels in IBM MQ for IBM i

Triggering of channels in IBM MQ for IBM i is implemented with the channel initiator process. A channel initiator process for the initiation queue SYSTEM.CHANNEL.INITQ is started automatically with the queue manager unless it is disabled by altering the queue manager SCHINIT attribute.

Set up the transmission queue for the channel, specifying SYSTEM.CHANNEL.INITQ as the initiation queue, and enabling triggering for the queue. The channel initiator starts the first available channel that specifies this transmission queue.

```
CRTMQMQ QNAME(MYXMITQ1) QTYPE(*LCL) MQMNAME(MYQMGR)
TRGENBL(*YES) INITQNAME(SYSTEM.CHANNEL.INITQ)
USAGE(*TMQ)
```

You can manually start up to three channel initiator processes with the STRMQMCHLI command and specify different initiation queues. You can also specify more than one channel able to process the transmission queue and choose which channel to start. This capability is still provided to be compatible with earlier releases. Its usage is deprecated.

Note: Only one channel at a time can process a transmission queue.

```
STRMQMCHLI QNAME(MYINITQ)
```

Set up the transmission queue for the channel, specifying TRGENBL(*YES) and, to choose which channel to attempt to start, specify the channel name in the TRIGDATA field. For example:

```
CRTMQMQ QNAME(MYXMITQ2) QTYPE(*LCL) MQMNAME(MYQMGR)
TRGENBL(*YES) INITQNAME(MYINITQ)
USAGE(*TMQ) TRIGDATA(MYCHANNEL)
```

Related concepts:

"Starting and stopping the channel initiator"
Triggering is implemented using the channel initiator process.

Related tasks:

"Configuring distributed queuing" on page 961
This section provides more detailed information about intercommunication between IBM MQ installations, including queue definition, channel definition, triggering, and sync point procedures

Related information:

Channel programs on UNIX, Linux, and Windows

IBM i Intercommunication jobs on IBM i

IBM i Channel states on IBM i

Starting and stopping the channel initiator:

Triggering is implemented using the channel initiator process.

This channel initiator process is started with the MQSC command START CHINIT. Unless you are using the default initiation queue, specify the name of the initiation queue on the command. For example, to use the START CHINIT command to start queue IQ for the default queue manager, enter:

```
START CHINIT INITQ(IQ)
```

By default, a channel initiator is started automatically using the default initiation queue, SYSTEM.CHANNEL.INITQ. If you want to start all your channel initiators manually, follow these steps:

1. Create and start the queue manager.
2. Alter the queue manager's SCHINIT property to MANUAL
3. End and restart the queue manager

In IBM 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 UNIX and Linux systems, and in the registry for Windows systems.

See IBM MQ Control commands for details of the run channel initiator command **`runmqchi`**, and the other control commands.

Stopping the channel initiator

The default channel initiator is started automatically when you start a queue manager. All channel initiators are stopped automatically when a queue manager is stopped.

Initialization and configuration files

The handling of channel initialization data depends on your IBM MQ platform.

z/OS

z/OS systems

In IBM MQ for z/OS, initialization and configuration information is specified using the ALTER QMGR MQSC command. If you put ALTER QMGR commands in the CSQINP2 initialization input data set, they are processed every time the queue manager is started.

To run MQSC commands such as START LISTENER every time you start the channel initiator, put them in the CSQINPX initialization input data set and specify the optional DD statement CSQINPX in the channel initiator started task procedure.

For further information about CSQINP2 and CSQINPX, see Customize the initialization input data sets, and ALTER QMGR.

Windows, IBM i IBM i, UNIX and Linux systems

In IBM MQ for Windows, IBM i IBM i, UNIX and Linux systems, there are *configuration files* to hold basic configuration information about the IBM MQ installation.

There are two configuration files: one applies to the machine, the other applies to an individual queue manager.

IBM MQ configuration file
This file holds information relevant to all the queue managers on the IBM MQ system. The file is called `mqs.ini`. It is fully described in the Administering for IBM MQ for Windows, and in Administering IBM i, and in UNIX and Linux systems.

Queue manager configuration file
This file holds configuration information relating to one particular queue manager. The file is called `qm.ini`.

It is created during queue manager creation and can hold configuration information relevant to any aspect of the queue manager. Information held in the file includes details of how the configuration of the log differs from the default in the IBM MQ configuration file.

The queue manager configuration file is held in the root of the directory tree occupied by the queue manager. For example, for the DefaultPath attributes, the queue manager configuration files for a queue manager called QMNAME would be:

For UNIX and Linux systems:

```
/var/mqm/qmgrs/QMNAME/qm.ini
```

An excerpt of a `qm.ini` file follows. It specifies that the TCP/IP listener is to listen on port 2500, the maximum number of current channels is to be 200, and the maximum number of active channels is to be 100.

```
TCP:
Port=2500
CHANNELS:
MaxChannels=200
MaxActiveChannels=100
```

You can specify a range of TCP/IP ports to be used by an outbound channel. One method is to use the qm.ini file to specify the start and end of a range of port values. The following example shows a qm.ini file specifying a range of channels:

```
TCP:
StrPort=2500
EndPort=3000
CHANNELS:
MaxChannels=200
MaxActiveChannels=100
```

If you specify a value for StrPort or EndPort then you must specify a value for both. The value of EndPort must always be greater than the value of StrPort.

The channel tries to use each of the port values in the range specified. When the connection is successful, the port value is the port that the channel then uses.

IBM i For IBM i:

```
/QIBM/UserData/mqm/qmgrs/QMNAME/qm.ini
```

For Windows systems:

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

For more information about `qm.ini` files, see Configuration file stanzas for distributed queuing.

Data conversion

IBM MQ messages might require data conversion when sent between queues on different queue managers.

An IBM MQ message consists of two parts:

- Control information in a message descriptor
- Application data

Either of the two parts might require data conversion when sent between queues on different queue managers. For information about application data conversion, see Application data conversion.

Writing your own message channel agents

IBM MQ allows you to write your own message channel agent (MCA) programs or to install one from an independent software vendor.

You might want to write your own MCA programs to make IBM MQ interoperate over your own proprietary communications protocol, or to send messages over a protocol that IBM MQ does not support. (You cannot write your own MCA to interoperate with an IBM MQ-supplied MCA at the other end.)

If you decide to use an MCA that was not supplied by IBM MQ, you must consider the following points.

Message sending and receiving
: You must write a sending application that gets messages from wherever your application puts them, for example from a transmission queue, and sends them out on a protocol with which you want to communicate. You must also write a receiving application that takes messages from this protocol and puts them onto destination queues. The sending and receiving applications use the message queue interface (MQI) calls, not any special interfaces.

: You must ensure that messages are only delivered once. Sync point coordination can be used to help with this delivery.

Channel control function
: You must provide your own administration functions to control channels. You cannot use IBM MQ channel administration functions either for configuring (for example, the DEFINE CHANNEL command) or monitoring (for example, DISPLAY CHSTATUS) your channels.

Initialization file
: You must provide your own initialization file, if you require one.

Application data conversion
: You probably want to allow for data conversion for messages you send to a different system. If so, use the MQGMO_CONVERT option on the MQGET call when retrieving messages from wherever your application puts them, for example the transmission queue.

User exits
: Consider whether you need user exits. If so, you can use the same interface definitions that IBM MQ uses.

Triggering
: If your application puts messages to a transmission queue, you can set up the transmission queue attributes so that your sending MCA is triggered when messages arrive on the queue.

Channel initiator
: You might must provide your own channel initiator.

Other things to consider for distributed queue management

Other topics to consider when preparing IBM MQ for distributed queue management. This topic covers Undelivered-message queue, Queues in use, System extensions and user-exit programs, and Running channels and listeners as trusted applications.

Undelivered-message queue

To ensure that messages arriving on the undelivered-message queue (also known as the dead-letter queue or DLQ) are processed, create a program that can be triggered or run at regular intervals to handle these messages.

UNIX Linux A DLQ handler is provided with IBM MQ on UNIX and Linux systems; for more information, see The sample DLQ handler, amqsdlq.

IBM i For more information on IBM MQ for IBM i, see The IBM MQ for IBM i dead-letter queue handler.

Queues in use

MCAs for receiver channels can keep the destination queues open even when messages are not being transmitted. This results in the queues appearing to be "in use".

Maximum number of channels

IBM i On IBM MQ for IBM i you can specify the maximum number of channels allowed in your system and the maximum number that can be active at one time. You specify these numbers in the `qm.ini` file in directory QIBM/UserData/mqm/qmgrs/*queue_manager_name*. See Configuration file stanzas for distributed queuing.

System extensions and user-exit programs

A facility is provided in the channel definition to enable extra programs to be run at defined times during the processing of messages. These programs are not supplied with IBM MQ, but can be provided by each installation according to local requirements.

In order to run, these user-exit programs must have predefined names and be available on call to the channel programs. The names of the user-exit programs are included in the message channel definitions.

There is a defined control block interface for handing over control to these programs, and for handling the return of control from these programs.

The precise places where these programs are called, and details of control blocks and names, are to be found in Channel-exit programs for messaging channels.

Running channels and listeners as trusted applications

If performance is an important consideration in your environment and your environment is stable, you can run your channels and listeners as trusted, using the FASTPATH binding. There are two factors that influence whether channels and listeners run as trusted:

- The environment variable MQ_CONNECT_TYPE=FASTPATH or MQ_CONNECT_TYPE=STANDARD. This is case-sensitive. If you specify a value that is not valid it is ignored.
- MQIBindType in the Channels stanza of the `qm.ini` or registry file. You can set this to FASTPATH or STANDARD and it is not case-sensitive. The default is STANDARD.

You can use MQIBindType in association with the environment variable to achieve the required effect as follows:

MQIBindType	Environment variable	Result
STANDARD	UNDEFINED	STANDARD
FASTPATH	UNDEFINED	FASTPATH
STANDARD	STANDARD	STANDARD
FASTPATH	STANDARD	STANDARD
STANDARD	FASTPATH	STANDARD
FASTPATH	FASTPATH	FASTPATH
STANDARD	CLIENT	CLIENT
FASTPATH	CLIENT	STANDARD
STANDARD	LOCAL	STANDARD
FASTPATH	LOCAL	STANDARD

In summary, there are only two ways of actually making channels and listeners run as trusted:

1. By specifying MQIBindType=FASTPATH in `qm.ini` or registry and not specifying the environment variable.
2. By specifying MQIBindType=FASTPATH in `qm.ini` or registry and setting the environment variable to FASTPATH.

Consider running listeners as trusted, because listeners are stable processes. Consider running channels as trusted, unless you are using unstable channel exits or the command STOP CHANNEL MODE(TERMINATE).

Monitoring and controlling channels on UNIX, Linux, and Windows

ULW

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 UNIX, 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 The control commands.

Programmable command format commands

For details, see PCF commands.

Linux Windows **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 117 on page 1017.

Procedure

- "Functions required for setting up and controlling channels"
- "Getting started with objects" on page 1018
- Windows "Setting up communication on Windows" on page 1025
- UNIX Linux "Setting up communication on UNIX and Linux" on page 1033

Related tasks:

IBM i "Monitoring and controlling channels on IBM i" on page 1039
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 information:

ULW Channel programs on UNIX, Linux, and Windows

ULW Message channel planning example for UNIX, Linux, and Windows

Example configuration information

Channel attributes

Functions required for setting up and controlling channels

ULW

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 1018
- "Creating associated objects" on page 1019

- "Creating default objects" on page 1019
- "Creating a channel" on page 1020
- "Displaying a channel" on page 1020
- "Displaying channel status" on page 1021
- "Checking links using Ping" on page 1021
- "Starting a channel" on page 1022
- "Stopping a channel" on page 1023
- "Renaming a channel" on page 1024
- "Resetting a channel" on page 1024
- "Resolving in-doubt messages on a channel" on page 1024

Table 117 shows the full list of IBM MQ functions that you might need.

Table 117. Functions required in UNIX, 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	dltmqm		Yes
Display queue manager		DISPLAY QMGR	Yes
End queue manager	endmqm		Yes
Ping queue manager		PING QMGR	No
Start queue manager	strmqm		Yes
Command server functions			
Display command server	dspmqcsv		No
End command server	endmqcsv		No
Start command server	strmqcsv		No
Queue functions			
Change queue		ALTER QALIAS ALTER QLOCAL ALTER QMODEL ALTER QREMOTE See, ALTER queues.	Yes
Clear queue		CLEAR QLOCAL	Yes
Create queue		DEFINE QALIAS DEFINE QLOCAL DEFINE QMODEL DEFINE QREMOTE See, DEFINE queues.	Yes
Delete queue		DELETE QALIAS DELETE QLOCAL DELETE QMODEL DELETE QREMOTE See, DELETE queues.	Yes
Display queue		DISPLAY QUEUE	Yes

Table 117. Functions required in UNIX, Linux, and Windows systems (continued)

Function	**Control commands**	**MQSC**	**IBM MQ Explorer equivalent?**
Process functions			
Change process		ALTER PROCESS	Yes
Create process		DEFINE PROCESS	Yes
Delete process		DELETE PROCESS	Yes
Display process		DISPLAY PROCESS	Yes
Channel functions			
Change channel		ALTER CHANNEL	Yes
Create channel		DEFINE CHANNEL	Yes
Delete channel		DELETE CHANNEL	Yes
Display channel		DISPLAY CHANNEL	Yes
Display channel status		DISPLAY CHSTATUS	Yes
End channel		STOP CHANNEL	Yes
Ping channel		PING CHANNEL	Yes
Reset channel		RESET CHANNEL	Yes
Resolve channel		RESOLVE CHANNEL	Yes
Run channel	runmqchl	START CHANNEL	Yes
Run channel initiator	runmqchi	START CHINIT	No
Run listener [1]	runmqlsr	START LISTENER	No
End listener	endmqlsr (Windows systems, AIX, HP-UX, and Solaris only)		No

Note:
1. A listener might be started automatically when the queue manager starts.

Getting started with objects

ULW

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: ULW

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 UNIX, Linux, and Windows.

Creating default objects: ULW

Default objects are created automatically when a queue manager is created. These objects are queues, channels, a process definition, and administration queues. After the default objects have been created, you can replace them at any time by running the strmqm command with the -c option.

When you use the crtmqm command to create a queue manager, the command also initiates a program to create a set of default objects.
1. Each default object is created in turn. The program keeps a count of how many objects are successfully defined, how many existed and were replaced, and how many unsuccessful attempts there were.
2. The program displays the results to you and if any errors occurred, directs you to the appropriate error log for details.

When the program has finished running, you can use the strmqm command to start the queue manager.

See The control commands for more information about the crtmqm and strmqm commands.

Changing the default objects

When you specify the -c option, the queue manager is started temporarily while the objects are created and is then shut down again. Issuing strmqm with the -c option refreshes existing system objects with the default values (for example, the MCAUSER attribute of a channel definition is set to blanks). You must use the strmqm command again, without the -c option, if you want to start the queue manager.

If you want to change the default objects, you can create your own version of the old amqscoma.tst file and edit it.

Creating a channel: ULW

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 UNIX, 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:

- Windows On Windows use Ctrl-z to end the entry at the command line.
- UNIX Linux On UNIX and Linux, use Ctrl-d.
- Alternatively, on UNIX, Linux, and Windows, use the **`end`** command.

Displaying a channel: ULW

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: ULW

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
- Windows 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: ULW

Use the MQSC command PING CHANNEL to exchange a fixed data message with the remote end.

Ping gives some confidence to the system supervisor that the link is available and functioning.

Ping does not involve the use of transmission queues and target queues. It uses channel definitions, the related communication link, and the network setup. It can only be used if the channel is not currently active.

It is available from sender, server, and cluster-sender channels only. The corresponding channel is started at the far side of the link, and performs the startup parameter negotiation. Errors are notified normally.

The result of the message exchange is presented as `Ping complete` or an error message.

Ping with LU 6.2

When Ping is invoked, by default no user ID or password flows to the receiving end. If user ID and password are required, they can be created at the initiating end in the channel definition. If a password is entered into the channel definition, it is encrypted by IBM MQ before being saved. It is then decrypted before flowing across the conversation.

Starting a channel: ULW

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:

- UNIX Linux For TCP on UNIX and Linux, inetd can be configured to start a channel. inetd is started as a separate process.
- UNIX Linux For LU 6.2 in UNIX and Linux, configure your SNA product to start the LU 6.2 responder process.
- Windows 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 **Windows**

MQ_INSTALLATION_PATH`\qmgrs\qmname\errors\AMQERR01.LOG` (for each queue manager called `qmname`)

MQ_INSTALLATION_PATH`\qmgrs\@SYSTEM\errors\AMQERR01.LOG` (for general errors)

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

Note: On Windows, you still also get a message in the Windows systems application event log.

UNIX Linux **UNIX 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 UNIX, Linux, and Windows, use the **runmqlsr** command to start the IBM MQ listener process. By default, any inbound requests for channel attachment causes the listener process to start MCAs as threads of the amqrmppa process.

```
runmqlsr -t tcp -m QM2
```

For outbound connections, you must start the channel in one of the following three ways:

1. Use the MQSC command START CHANNEL, specifying the channel name, to start the channel as a process or a thread, depending on the MCATYPE parameter. (If channels are started as threads, they are threads of a channel initiator.)
   ```
   START CHANNEL(QM1.TO.QM2)
   ```
2. Use the control command runmqchl to start the channel as a process.
   ```
   runmqchl -c QM1.TO.QM2 -m QM1
   ```
3. Use the channel initiator to trigger the channel.

Stopping a channel: ULW

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 1003.

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: ULW

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: ULW

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 1018
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 991
The channel control function provides facilities for you to define, monitor, and control channels.

Related tasks:
"Configuring distributed queuing" on page 961
This section provides more detailed information about intercommunication between IBM MQ installations, including queue definition, channel definition, triggering, and sync point procedures

Related information:
RESET CHANNEL

Resolving in-doubt messages on a channel: ULW

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 1018
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 991
The channel control function provides facilities for you to define, monitor, and control channels.

Related tasks:

"Configuring distributed queuing" on page 961
This section provides more detailed information about intercommunication between IBM MQ installations, including queue definition, channel definition, triggering, and sync point procedures

Related information:

RESOLVE CHANNEL

Setting up communication on Windows

Windows

When a distributed-queuing management channel is started, it tries to use the connection specified in the channel definition. For this to succeed, it is necessary for the connection to be defined and available. This section explains how to do this 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.

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 1026
- "Defining an LU 6.2 connection on Windows" on page 1028
- "Defining a NetBIOS connection on Windows" on page 1030

Related tasks:

"Monitoring and controlling channels on UNIX, Linux, and Windows" on page 1015
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 server and client" on page 844
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 UNIX and Linux" on page 1033
DQM is a remote queuing facility for IBM MQ. It provides channel control programs for the queue manager which form the interface to communication links, controllable by the system operator. The channel definitions held by distributed-queuing management use these connections.

Related reference:

"Which communication type to use" on page 845
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: Windows

Define a TCP connection by configuring a channel at the sending end to specify the address of the target, and by running a listener program at the receiving end.

Sending end

Specify the host name, or the TCP address of the target machine, in the Connection name field of the channel definition.

The port to connect to defaults to 1414. Port number 1414 is assigned by the Internet Assigned Numbers Authority to IBM MQ.

To use a port number other than the default, specify it in the connection name field of the channel object definition thus:

```
DEFINE CHANNEL('channel name') CHLTYPE(SDR) +
         TRPTYPE(TCP) +
         CONNAME('OS2ROG3(1822)') +
         XMITQ('XMitQ name') +
         REPLACE
```

where `OS2ROG3` is the DNS name of the remote queue manager and `1822` is the port required. (This must be the port that the listener at the receiving end is listening on.)

A running channel must be stopped and restarted to pick up any change to the channel object definition.

You can change the default port number by specifying it in the `.ini` file for IBM MQ for Windows:

```
TCP:
Port=1822
```

Note: To select which TCP/IP port number to use, IBM MQ uses the first port number it finds in the following sequence:

1. The port number explicitly specified in the channel definition or command line. This number allows the default port number to be overridden for a channel.
2. The port attribute specified in the TCP stanza of the `.ini` file. This number allows the default port number to be overridden for a queue manager.

3. The default value of 1414. This is the number assigned to IBM MQ by the Internet Assigned Numbers Authority for both inbound and outbound connections.

For more information about the values you set using qm.ini, see Configuration file stanzas for distributed queuing.

Receiving on TCP

To start a receiving channel program, a listener program must be started to detect incoming network requests and start the associated channel. You can use the IBM MQ listener.

Receiving channel programs are started in response to a startup request from the sending channel.

To start a receiving channel program, a listener program must be started to detect incoming network requests and start the associated channel. You can use the IBM MQ listener.

To run the Listener supplied with IBM MQ, that starts new channels as threads, use the runmqlsr command.

A basic example of using the **runmqlsr** command:

```
runmqlsr -t tcp [-m QMNAME] [-p 1822]
```

The square brackets indicate optional parameters; QMNAME is not required for the default queue manager, and the port number is not required if you are using the default (1414). The port number must not exceed 65535.

Note: To select which TCP/IP port number to use, IBM MQ uses the first port number it finds in the following sequence:
1. The port number explicitly specified in the channel definition or command line. This number allows the default port number to be overridden for a channel.
2. The port attribute specified in the TCP stanza of the `.ini` file. This number allows the default port number to be overridden for a queue manager.
3. The default value of 1414. This is the number assigned to IBM MQ by the Internet Assigned Numbers Authority for both inbound and outbound connections.

For the best performance, run the IBM MQ listener as a trusted application as described in "Running channels and listeners as trusted applications" on page 1014. 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 1000.

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.

Defining an LU 6.2 connection on Windows: 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 118. Settings on the local Windows system for a remote queue manager platform

Remote platform	TPNAME	TPPATH
z/OS or MVS/ESA without CICS	The same as in the corresponding side information about the remote queue manager.	-
z/OS or MVS/ESA using CICS	CKRC (sender) CKSV (requester) CKRC (server)	-
IBM i	The same as the compare value in the routing entry on the IBM i system.	-
UNIX and Linux systems	The same as in the corresponding side information about the remote queue manager.	*MQ_INSTALLATION_PATH*/bin/amqcrs6a
Windows	As specified in the Windows Run Listener command, or the invokable Transaction Program that was defined using TpSetup on Windows.	*MQ_INSTALLATION_PATH*\bin\amqcrs6a

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

If you have more than one queue manager on the same machine, ensure that the TPnames in the channel definitions are unique.

For the latest information about configuring AnyNet® SNA over TCP/IP, see the following online IBM documentation: AnyNet SNA over TCP/IP and SNA Node Operations.

Related concepts:

"Sending end on LU 6.2 on 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.

"Receiving on LU 6.2 on Windows" on page 1029
Receiving channel programs are started in response to a startup request from the sending channel.

Sending end on LU 6.2 on Windows: 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          OS2ROG2
Partner TP Name          recv
Mode Name                #INTER
```

Receiving on LU 6.2 on Windows: 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 last part in square brackets is optional and is not required for the default queue manager.

It is possible to have more than one queue manager running on one machine. You must assign a different TpName to each queue manager, and then start a listener program for each one. For example:

```
RUNMQLSR -t LU62 -m QM1 -n TpName1
RUNMQLSR -t LU62 -m QM2 -n TpName2
```

For the best performance, run the IBM MQ listener as a trusted application as described in Running channels and listeners as trusted applications. See Restrictions for trusted applications for information about trusted applications.

You can stop all IBM MQ listeners running on a queue manager that is inactive, using the command:

```
ENDMQLSR [-m QMNAME]
```

Using Microsoft SNA Server on Windows

You can use TpSetup (from the SNA Server SDK) to define an invokable TP that then drives amqcrs6a.exe, or you can set various registry values manually. The parameters that should be passed to amqcrs6a.exe are:

```
-m QM -n TpName
```

where *QM* is the Queue Manager name and *TpName* is the TP Name. See the *Microsoft SNA Server APPC Programmers Guide* or the *Microsoft SNA Server CPI-C Programmers Guide* for more information.

If you do not specify a queue manager name, the default queue manager is assumed.

Defining a NetBIOS connection on Windows: ▶ 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"
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 1031
The queue manager limits for NetBIOS sessions, commands, and names can be specified in two ways.

"Establishing the LAN adapter number" on page 1031
For channels to work successfully across NetBIOS, the adapter support at each end must be compatible. IBM MQ allows you to control the choice of LAN adapter (LANA) number by using the AdapterNum value in the NETBIOS stanza of your qm.ini file and by specifying the **-a** parameter on the runmqlsr command.

"Initiating the NetBIOS connection" on page 1032
Defining the steps needed to initiate a connection.

"Defining the target listener for the NetBIOS connection" on page 1032
Defining the steps to be undertaken at the receiving end of the NetBIOS connection.

Defining the IBM MQ local NetBIOS name: ▶ Windows

The local NetBIOS name used by IBM MQ channel processes can be specified in three ways.

In order of precedence the three ways are:

1. The value specified in the **-l** parameter of the RUNMQLSR command, for example:
   ```
   RUNMQLSR -t NETBIOS -l my_station
   ```
2. The MQNAME environment variable with a value that is established by the command:
   ```
   SET MQNAME= my_station
   ```
   You can set the MQNAME value for each process. Alternatively, you can set it at a system level in the Windows registry.

   If you are using a NetBIOS implementation that requires unique names, you must issue a SET MQNAME command in each window in which an IBM MQ process is started. The MQNAME value is arbitrary but it must be unique for each process.
3. The NETBIOS stanza in the queue manager configuration file qm.ini. For example:

```
NETBIOS:

LocalName= my_station
```

Note:

1. Due to the variations in implementation of the NetBIOS products supported, you are advised to make each NetBIOS name unique in the network. If you do not, unpredictable results might occur. If you have problems establishing a NetBIOS channel and there are error messages in the queue manager error log showing a NetBIOS return code of X'15', review your use of NetBIOS names.
2. On Windows, you cannot use your machine name as the NetBIOS name because Windows already uses it.
3. Sender channel initiation requires that a NetBIOS name be specified either by using the MQNAME environment variable or the LocalName in the qm.ini file.

Establishing the queue manager NetBIOS session, command, and name limits: Windows

The queue manager limits for NetBIOS sessions, commands, and names can be specified in two ways.

In order of precedence these ways are:

1. The values specified in the RUNMQLSR command:
   ```
   -s Sessions
   -e Names
   -o Commands
   ```
   If the -m operand is not specified in the command, the values apply only to the default queue manager.
2. The NETBIOS stanza in the queue manager configuration file qm.ini. For example:
   ```
   NETBIOS:

   NumSess= Qmgr_max_sess
   NumCmds= Qmgr_max_cmds
   NumNames= Qmgr_max_names
   ```

Establishing the LAN adapter number: Windows

For channels to work successfully across NetBIOS, the adapter support at each end must be compatible. IBM MQ allows you to control the choice of LAN adapter (LANA) number by using the AdapterNum value in the NETBIOS stanza of your qm.ini file and by specifying the **`-a`** parameter on the runmqlsr command.

The default LAN adapter number used by IBM MQ for NetBIOS connections is 0. Verify the number being used on your system as follows:

On Windows, it is not possible to query the LAN adapter number directly through the operating system. Instead, you use the LANACFG.EXE command-line utility, available from Microsoft. The output of the tool shows the virtual LAN adapter numbers and their effective bindings. For further information about LAN adapter numbers, see the Microsoft Knowledge Base article 138037 *HOWTO: Use LANA Numbers in a 32-bit Environment*.

Specify the correct value in the NETBIOS stanza of the queue manager configuration file, qm.ini:

```
NETBIOS:
AdapterNum= n
```

where n is the correct LAN adapter number for this system.

Initiating the NetBIOS connection: Windows

Defining the steps needed to initiate a connection.

To initiate the connection, follow these steps at the sending end:
1. Define the NetBIOS station name using the MQNAME or LocalName value.
2. Verify the LAN adapter number being used on your system and specify the correct file using the AdapterNum.
3. In the ConnectionName field of the channel definition, specify the NetBIOS name being used by the target listener program. On Windows, NetBIOS channels must be run as threads. Do this by specifying MCATYPE(THREAD) in the channel definition.
   ```
   DEFINE CHANNEL (chname) CHLTYPE(SDR) +
   TRPTYPE(NETBIOS) +
   CONNAME(your_station) +
   XMITQ(xmitq) +
   MCATYPE(THREAD) +
   REPLACE
   ```

Defining the target listener for the NetBIOS connection: Windows

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 1014. See Restrictions for trusted applications for information about trusted applications.

You can stop all IBM MQ listeners running on a queue manager that is inactive, using the command:
```
ENDMQLSR [-m QMNAME]
```

If you do not specify a queue manager name, the default queue manager is assumed.

Setting up communication on UNIX and Linux

UNIX Linux

DQM is a remote queuing facility for IBM MQ. It provides channel control programs for the queue manager which form the interface to communication links, controllable by the system operator. The channel definitions held by distributed-queuing management use these connections.

Before you begin

You might find it helpful to refer to the following sections:

- AIX Example configuration - IBM MQ for AIX
- HP-UX Example configuration - IBM MQ for HP-UX
- Solaris Example configuration - IBM MQ for Solaris
- Linux Example configuration - IBM MQ for Linux

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 UNIX 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. For more information about IBM MQ MQI clients, see Overview of IBM MQ MQI clients.

Procedure

For information on setting up communication for your UNIX and Linux system, see the subtopic for your chosen communication type:
- “Defining a TCP connection on UNIX and Linux” on page 1034
- “Defining an LU 6.2 connection on UNIX and Linux” on page 1038

Related tasks:

"Monitoring and controlling channels on UNIX, Linux, and Windows" on page 1015
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 server and client" on page 844
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 1025
When a distributed-queuing management channel is started, it tries to use the connection specified in the channel definition. For this to succeed, it is necessary for the connection to be defined and available. This section explains how to do this by using the forms of communication available for IBM MQ for Windows systems.

Related reference:

"Which communication type to use" on page 845
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 UNIX and Linux: UNIX Linux

The channel definition at the sending end specifies the address of the target. The listener or inet daemon is configured for the connection at the receiving end.

Sending end

Specify the host name, or the TCP address of the target machine, in the Connection Name field of the channel definition. The port to connect to defaults to 1414. Port number 1414 is assigned by the Internet Assigned Numbers Authority to IBM MQ.

To use a port number other than the default, change the connection name field thus:

```
Connection Name REMHOST(1822)
```

where `REMHOST` is the host name of the remote machine and 1822 is the port number required. (This must be the port that the listener at the receiving end is listening on.)

Alternatively you can change the port number by specifying it in the queue manager configuration file (qm.ini):

```
TCP:
Port=1822
```

For more information about the values you set using qm.ini, see Configuration file stanzas for distributed queuing.

Receiving on TCP

You can use either the TCP/IP listener, which is the inet daemon (inetd), or the IBM MQ listener.

Some Linux distributions now use the extended inet daemon (xinetd) instead of the inet daemon. For information about how to use the extended inet daemon on a Linux system, see Establishing a TCP connection on Linux.

Related concepts:

"Using the TCP/IP listener on UNIX and Linux"
To start channels on UNIX and Linux, the `/etc/services` file and the `inetd.conf` file must be edited

"Using the TCP listener backlog option on UNIX and Linux" on page 1036
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 1037
To run the listener supplied with IBM MQ, which starts new channels as threads, use the `runmqlsr` command.

"Using the TCP/IP SO_KEEPALIVE option" on page 1037
On some UNIX and Linux systems, you can define how long TCP waits before checking that the connection is still available, and how frequently it tries the connection again if the first check fails. This is either a kernel tunable parameter, or can be entered at the command line.

Using the TCP/IP listener on UNIX and Linux: UNIX Linux

To start channels on UNIX and Linux, the `/etc/services` file and the `inetd.conf` file must be edited

Follow these instructions:

1. Edit the `/etc/services` file:

   Note: To edit the `/etc/services` file, you must be logged in as a superuser or root. You can change this, but it must match the port number specified at the sending end.
   Add the following line to the file:

   ```
   MQSeries 1414/tcp
   ```

   where 1414 is the port number required by IBM MQ. The port number must not exceed 65535.
2. Add a line in the `inetd.conf` file to call the program amqcrsta, where *`MQ_INSTALLATION_PATH`* represents the high-level directory in which IBM MQ is installed:

   ```
   MQSeries stream tcp nowait mqm MQ_INSTALLATION_PATH/bin/amqcrsta amqcrsta
   [-m Queue_Man_Name]
   ```

The updates are active after inetd has reread the configuration files. To do this, issue the following commands from the root user ID:

- On AIX:
  ```
  refresh -s inetd
  ```
- On HP-UX, from the mqm user ID:
  ```
  inetd -c
  ```
- On Solaris 10 or later:
  ```
  inetconv
  ```
- On other UNIX and Linux systems (including Solaris 9):
  ```
  kill -1 process_number
  ```

When the listener program started by inetd inherits the locale from inetd, it is possible that the MQMDE is not honored (merged) and is placed on the queue as message data. To ensure that the MQMDE is honored, you must set the locale correctly. The locale set by inetd might not match that chosen for other locales used by IBM MQ processes. To set the locale:

1. Create a shell script which sets the locale environment variables LANG, LC_COLLATE, LC_CTYPE, LC_MONETARY, LC_NUMERIC, LC_TIME, and LC_MESSAGES to the locale used for other IBM MQ process.

2. In the same shell script, call the listener program.
3. Modify the `inetd.conf` file to call your shell script in place of the listener program.

It is possible to have more than one queue manager on the server. You must add a line to each of the two files, for each of the queue managers. For example:

```
MQSeries1   1414/tcp
MQSeries2   1822/tcp
```

```
MQSeries2 stream tcp nowait mqm MQ_INSTALLATION_PATH/bin/amqcrsta amqcrsta -m QM2
```

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

This avoids error messages being generated if there is a limitation on the number of outstanding connection requests queued at a single TCP port. For information about the number of outstanding connection requests, see "Using the TCP listener backlog option on UNIX and Linux."

Using the TCP listener backlog option on UNIX and Linux: UNIX Linux

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 119.

Table 119. Maximum outstanding connection requests queued at a TCP/IP port

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

If the backlog reaches the values shown in Table 119, 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 119) 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 `runmqlsr -b` command, or
- Use the MQSC command **`DEFINE LISTENER`** with the BACKLOG attribute set to the required value.

For information about the **`runmqlsr`** command, see runmqlsr. For information about the DEFINE LISTENER command, see the DEFINE LISTENER.

Using the IBM MQ listener: UNIX Linux

To run the listener supplied with IBM MQ, which starts new channels as threads, use the `runmqlsr` command.

For example:

```
runmqlsr -t tcp [-m QMNAME] [-p 1822]
```

The square brackets indicate optional parameters; `QMNAME` is not required for the default queue manager, and the port number is not required if you are using the default (1414). The port number must not exceed 65535.

For the best performance, run the IBM MQ listener as a trusted application as described in "Running channels and listeners as trusted applications" on page 1014. See Restrictions for trusted applications for information about trusted applications.

You can stop all IBM MQ listeners running on a queue manager that is inactive, using the command:

```
endmqlsr [-m QMNAME]
```

If you do not specify a queue manager name, the default queue manager is assumed.

Using the TCP/IP SO_KEEPALIVE option: UNIX Linux

On some UNIX and Linux systems, you can define how long TCP waits before checking that the connection is still available, and how frequently it tries the connection again if the first check fails. This is either a kernel tunable parameter, or can be entered at the command line.

If you want to use the SO_KEEPALIVE option (for more information, see "Checking that the other end of the channel is still available" on page 1000) you must add the following entry to your queue manager configuration file (qm.ini):

```
TCP:
KeepAlive=yes
```

See the documentation for your UNIX and Linux system for more information.

Defining an LU 6.2 connection on UNIX and Linux: UNIX 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 120. Settings on the local UNIX and Linux system for a remote queue manager platform

Remote platform	TPNAME	TPPATH
z/OS without CICS	The same as the corresponding TPName in the side information about the remote queue manager.	-
z/OS using CICS	CKRC (sender) CKSV (requester) CKRC (server)	-
IBM i	The same as the compare value in the routing entry on the IBM i system.	-
UNIX and Linux systems	The same as the corresponding TPName in the side information about the remote queue manager.	`MQ_INSTALLATION_PATH`/bin/amqcrs6a
Windows	As specified in the Windows Run Listener command, or the invokable Transaction Program that was defined using TpSetup on Windows.	`MQ_INSTALLATION_PATH`\bin\amqcrs6a

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

If you have more than one queue manager on the same machine, ensure that the TPnames in the channel definitions are unique.

Related concepts:

"Sending end on LU 6.2 on UNIX and Linux"
On UNIX and Linux systems, create a CPI-C side object (symbolic destination) and enter this name in the Connection name field in the channel definition. Also create an LU 6.2 link to the partner.

"Receiving on LU 6.2 on UNIX and Linux" on page 1039
On UNIX and Linux systems, create a listening attachment at the receiving end, an LU 6.2 logical connection profile, and a TPN profile.

Sending end on LU 6.2 on UNIX and Linux: UNIX Linux

On UNIX and Linux systems, create a CPI-C side object (symbolic destination) and enter this name in the Connection name field in the channel definition. Also create an LU 6.2 link to the partner.

In the CPI-C side object enter the partner LU name at the receiving machine, the transaction program name and the mode name. For example:

```
Partner LU Name             REMHOST
Remote TP Name              recv
Service Transaction Program no
Mode Name                   #INTER
```

On HP-UX, use the APPCLLU environment variable to name the local LU that the sender should use. On Solaris, set the APPC_LOCAL_LU environment variable to be the local LU name.

SECURITY PROGRAM is used, where supported by CPI-C, when IBM MQ attempts to establish an SNA session.

Receiving on LU 6.2 on UNIX and Linux: UNIX Linux

On UNIX and Linux systems, create a listening attachment at the receiving end, an LU 6.2 logical connection profile, and a TPN profile.

In the TPN profile, enter the full path to the executable file and the Transaction Program name:

```
Full path to TPN executable    MQ_INSTALLATION_PATH/bin/amqcrs6a
Transaction Program name      recv
User ID                       0
```

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

On systems where you can set the user ID, specify a user who is a member of the mqm group. On AIX, Solaris, and HP-UX, set the APPCTPN (transaction name) and APPCLLU (local LU name) environment variables (you can use the configuration panels for the invoked transaction program).

You might need to use a queue manager other than the default queue manager. If so, define a command file that calls:

```
amqcrs6a -m Queue_Man_Name
```

then call the command file.

Monitoring and controlling channels on IBM i

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 961.

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 1041
- “Creating a channel on IBM i” on page 1041
- “Starting a channel on IBM i” on page 1044
- “Selecting a channel on IBM i” on page 1044
- “Browsing a channel on IBM i” on page 1045
- “Renaming a channel on IBM i” on page 1047
- “Work with channel status on IBM i” on page 1047
- “Work-with-channel choices on IBM i” on page 1048

Related concepts:

“Setting up communication for IBM i” on page 1055
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 server and client” on page 844
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 information:

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

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 IBM MQ for IBM i system and default objects.

Creating a channel on IBM i

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.

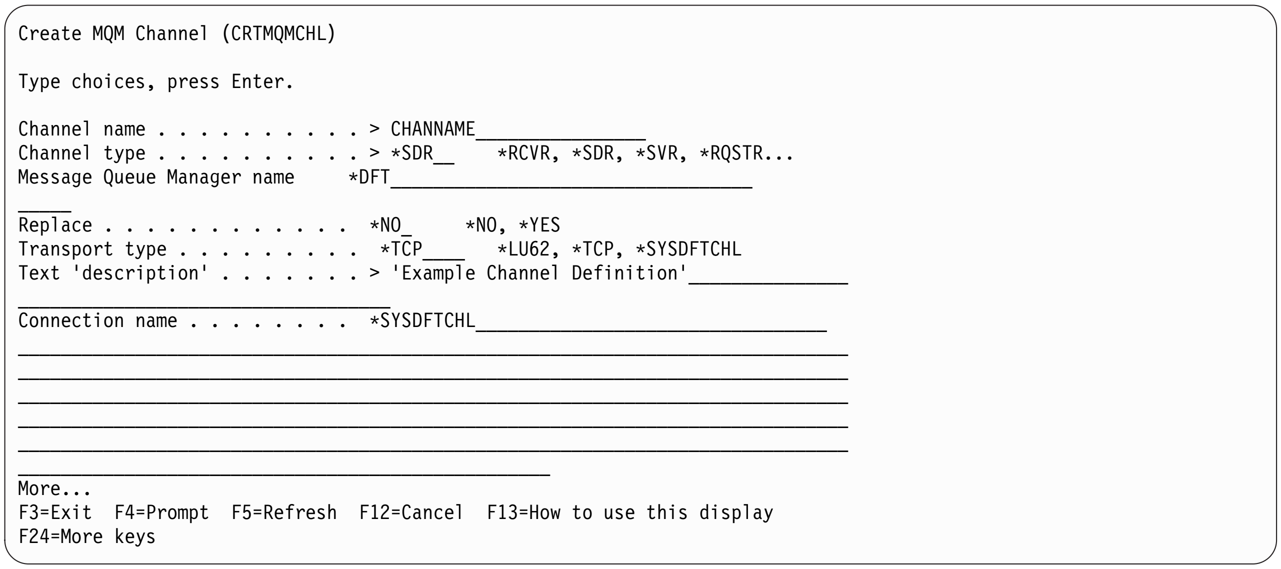
   Either way, the Create Channel panel is displayed. Type:
   - The name of the channel in the field provided
   - The channel type for this end of the link
2. Press enter.

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.

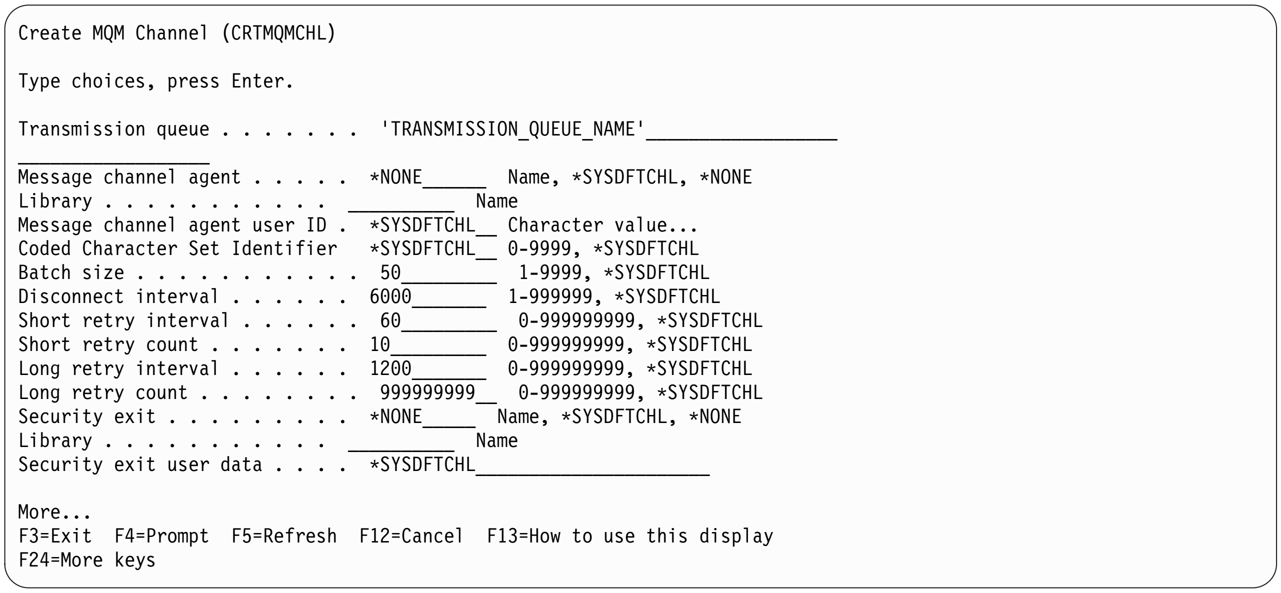
You are presented with the appropriate channel settings panel for the type of channel you have chosen. Complete the fields with the information you have gathered previously. Press enter to create the channel.

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.



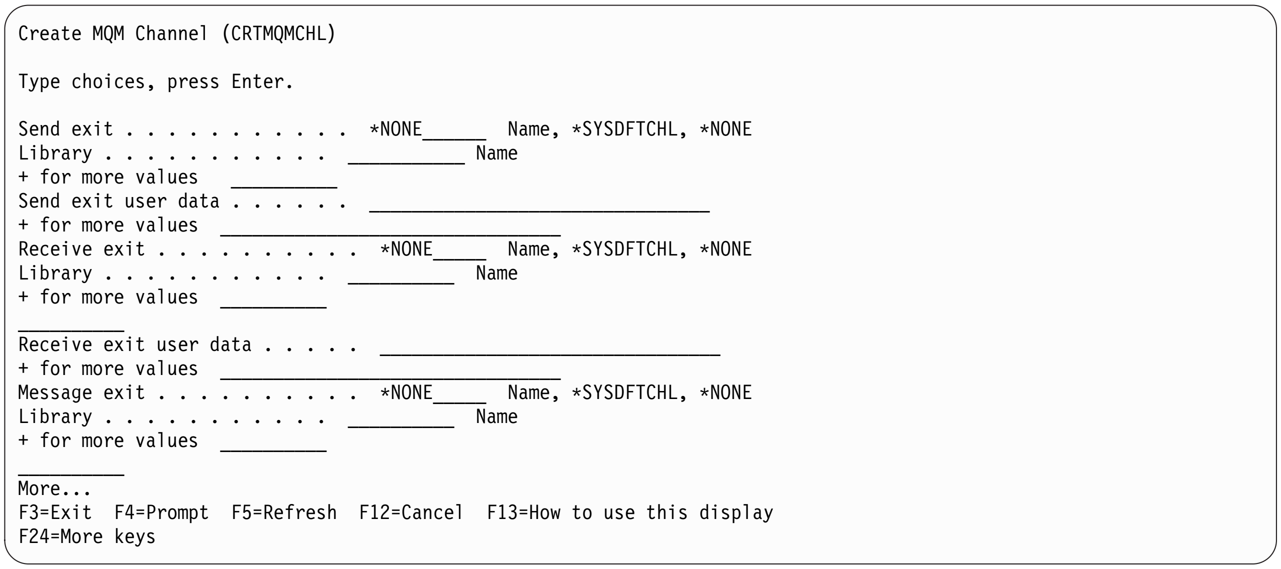
```
Create MQM Channel (CRTMQMCHL)

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 103. Create channel (1)



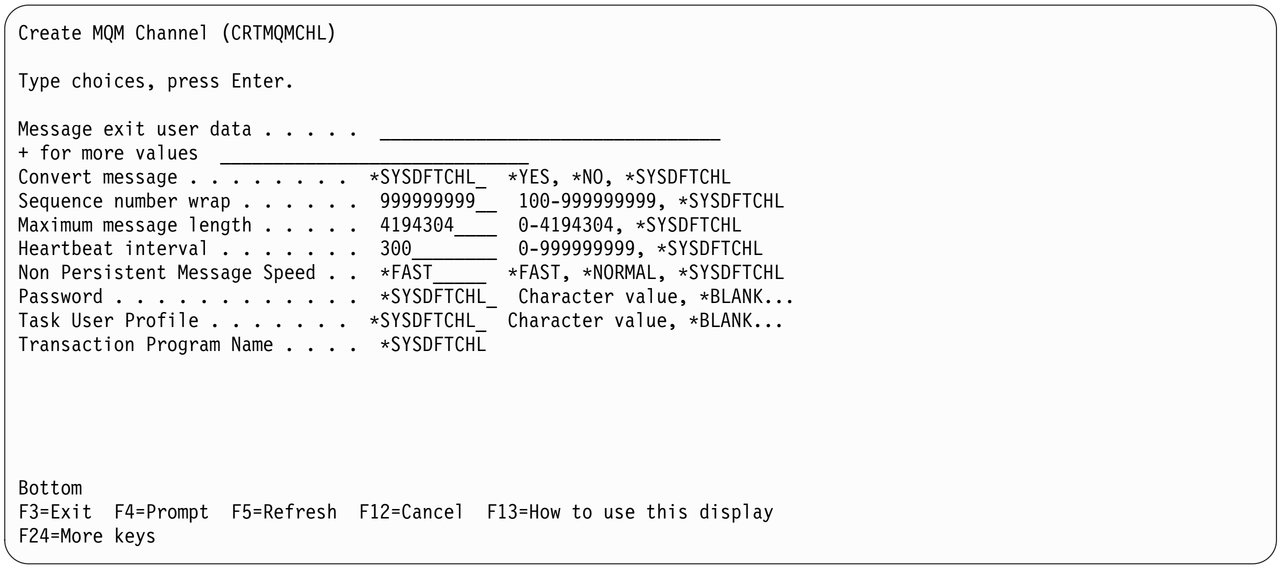
```
Create MQM Channel (CRTMQMCHL)

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 104. 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 105. Create channel (3)



```
Create MQM Channel (CRTMQMCHL)

Type choices, press Enter.

Message exit user data . . . . .  ________________________________
+ for more values  _____________________________
Convert message . . . . . . . .  *SYSDFTCHL_  *YES, *NO, *SYSDFTCHL
Sequence number wrap . . . . . .  999999999__  100-999999999, *SYSDFTCHL
Maximum message length . . . . .  4194304____  0-4194304, *SYSDFTCHL
Heartbeat interval . . . . . . .  300________  0-999999999, *SYSDFTCHL
Non Persistent Message Speed . .  *FAST_____  *FAST, *NORMAL, *SYSDFTCHL
Password . . . . . . . . . . . .  *SYSDFTCHL_  Character value, *BLANK...
Task User Profile . . . . . . .  *SYSDFTCHL_  Character value, *BLANK...
Transaction Program Name . . . .  *SYSDFTCHL





Bottom
F3=Exit  F4=Prompt  F5=Refresh  F12=Cancel  F13=How to use this display
F24=More keys
```

Figure 106. Create channel (4)

Starting a channel on IBM i

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.

Selecting a channel on IBM i

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 107. 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.

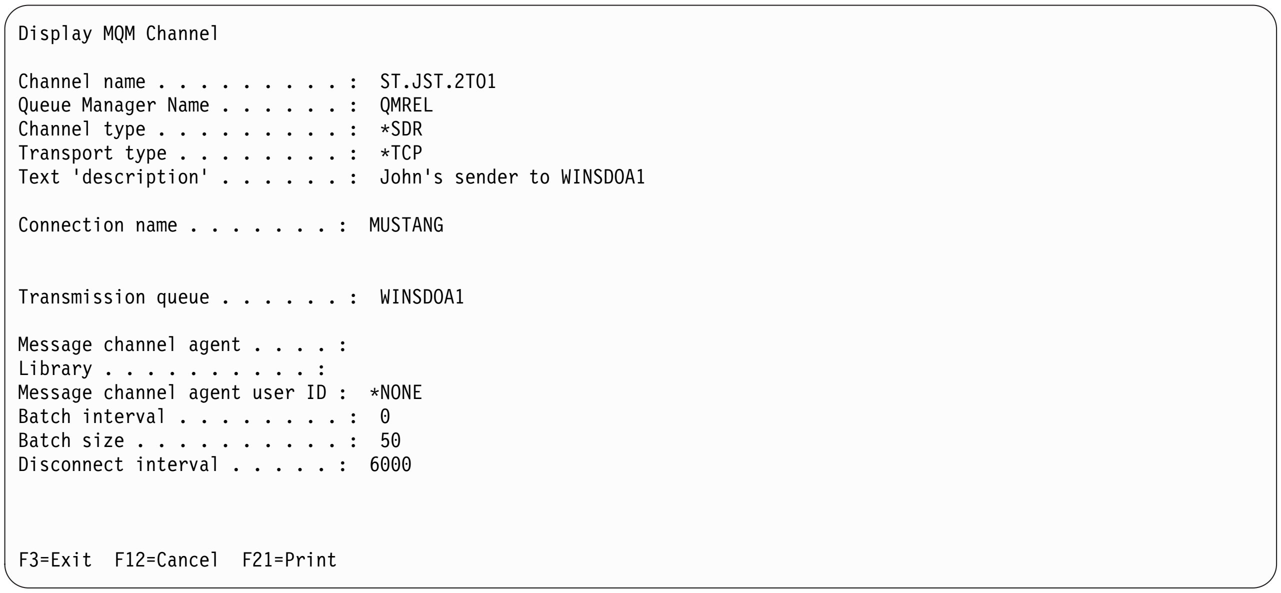
To browse the settings of a channel, use the WRKMQMCHL command to begin at the Display Channel panel:

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.

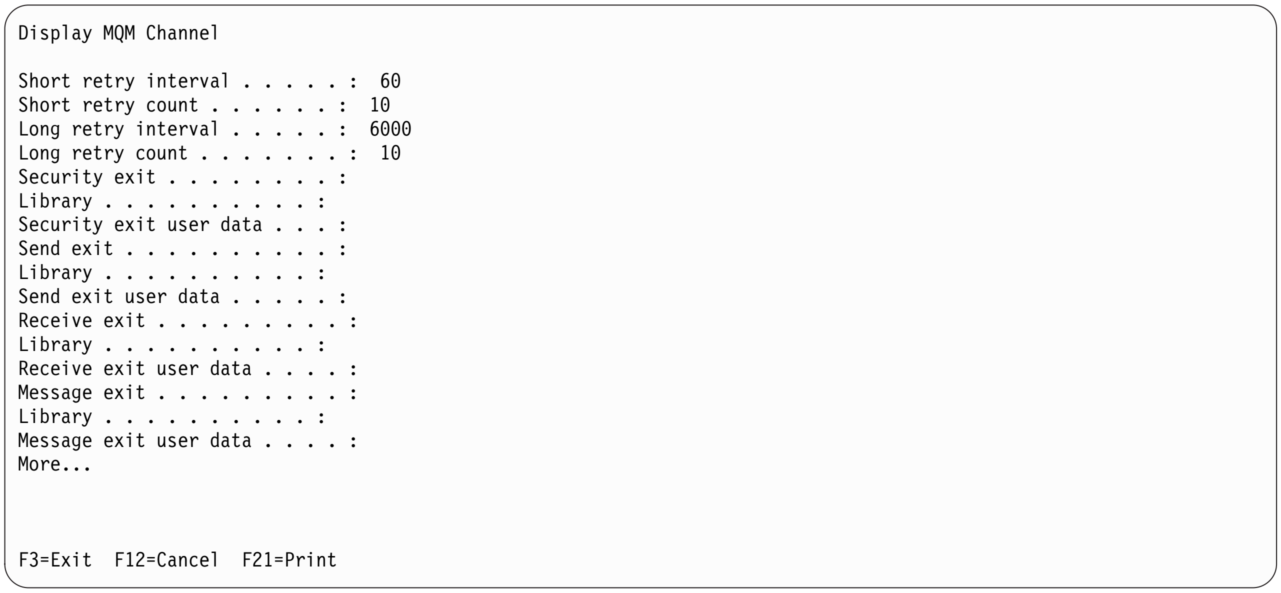
Alternatively, you can use the DSPMQMCHL command from the command line.

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.



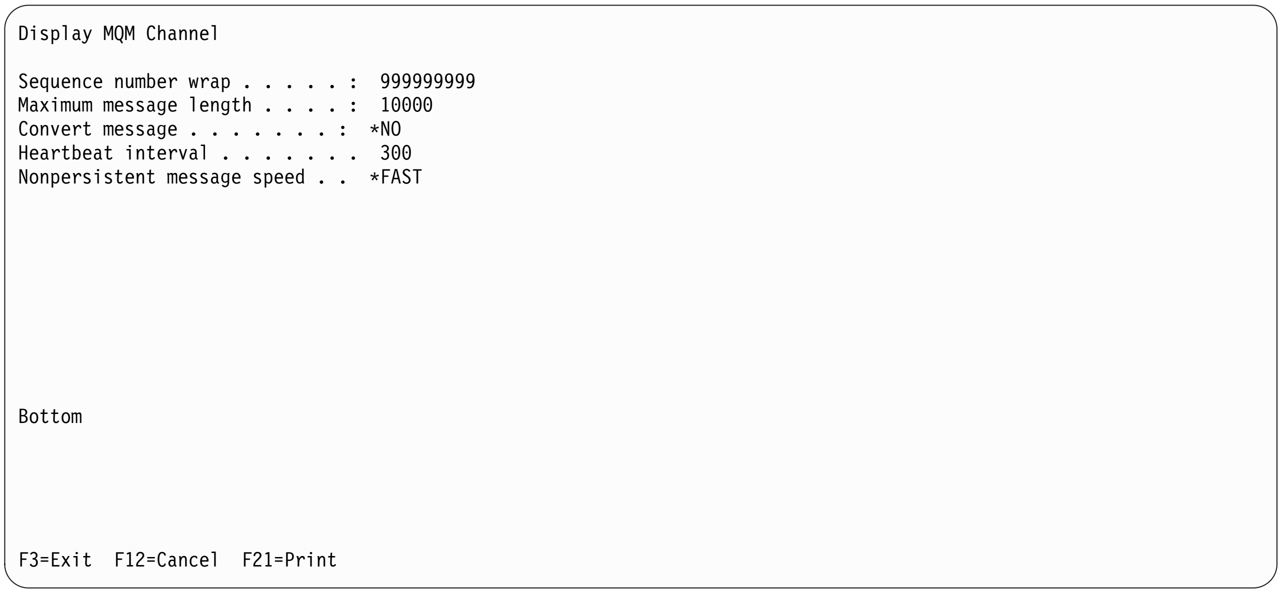
```
Display MQM Channel

Channel name . . . . . . . . . :  ST.JST.2TO1
Queue Manager Name . . . . . . :  QMREL
Channel type . . . . . . . . . :  *SDR
Transport type . . . . . . . . :  *TCP
Text 'description' . . . . . . :  John's sender to WINSDOA1

Connection name . . . . . . . :  MUSTANG


Transmission queue . . . . . . :  WINSDOA1

Message channel agent . . . . :
Library . . . . . . . . . . :
Message channel agent user ID :  *NONE
Batch interval . . . . . . . . :  0
Batch size . . . . . . . . . . :  50
Disconnect interval . . . . . :  6000



F3=Exit  F12=Cancel  F21=Print
```

Figure 108. 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 109. 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 110. Display a TCP/IP channel (3)

Renaming a channel on IBM i

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

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.2TO1.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 111. First of the set of channel status panels

The options available in the Work with Channel Status panel are:

Menu option	**Description**
5=Display	Displays the channel settings.
13=Ping	Initiates a Ping action, where appropriate.
14=Start	Starts the channel.
15=End	Stops the channel.
16=Reset	Resets the channel sequence number.
17=Resolve	Resolves an in-doubt channel situation, manually.

Work-with-channel choices on IBM i

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 1049	Changes the attributes of a channel.
“3=Copy” on page 1049	Copies the attributes of a channel to a new channel.
“4=Delete” on page 1049	Deletes a channel.
“5=Display” on page 1049	Displays the current settings for the channel.
“6=Create” on page 1050	Displays the Create channel panel
“8=Work with Status” on page 1050	Displays the channel status panels.
“13=Ping” on page 1051	Runs the Ping facility to test the connection to the adjacent system by exchanging a fixed data message with the remote end.
“14=Start” on page 1051	Starts the selected channel, or resets a disabled receiver channel.
“15=End” on page 1052	Requests the channel to close down.
“16=Reset” on page 1053	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.

Menu option	Description
"17=Resolve" on page 1053	Requests the channel to resolve in-doubt messages without establishing connection to the other end.
"18=Display authority" on page 1054	Displays IBM MQ object authority
"19=Grant authority" on page 1054	Grants IBM MQ object authority
"20=Revoke authority" on page 1054	Revokes IBM MQ object authority
"21=Recover object" on page 1054	Recovers IBM MQ object
"22=Record image" on page 1054	Records IBM MQ object image

2=Change: IBM i

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: IBM i

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: IBM i

Use the Delete option to delete the selected channel.

A panel is displayed to confirm or cancel your request.

5=Display: IBM i

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: IBM i

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 103 on page 1042.

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 information:

Channel attributes

8=Work with Status: IBM i

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 1047.

- 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: IBM i

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: IBM i

Use the Start option to start a channel manually.

The Start option is available for sender, server, and requester channels. It is not necessary where a channel has been set up with queue manager triggering.

The Start option is also used for receiver, server-connection, cluster sender, and cluster receiver channels. Starting a receiver channel that is in STOPPED state means that it can be started from the remote channel.

When started, the sending MCA reads the channel definition file and opens the transmission queue. A channel start-up sequence is issued, which remotely starts the corresponding MCA of the receiver or server channel. When they have been started, the sender and server processes await messages arriving on the transmission queue and transmit them as they arrive.

When you use triggering, you must start the continuously running trigger process to monitor the initiation queue. The STRMQMCHLI command can be used for starting the process.

At the far end of a channel, the receiving process might be started in response to a channel startup from the sending end. The method of doing so is different for LU 6.2 and TCP/IP connected channels:

- LU 6.2 connected channels do not require any explicit action at the receiving end of a channel.
- TCP connected channels require a listener process to be running continuously. This process awaits channel startup requests from the remote end of the link and starts the process defined in the channel definitions for that connection.

  When the remote system is IBM i, you can use the STRMQMLSR command.

Use of the Start option always causes the channel to resynchronize, where necessary.

For the start to succeed:

- Channel definitions, local and remote must exist. If there is no appropriate channel definition for a receiver or server-connection channel, a default one is created automatically if the channel is auto-defined. See Channel auto-definition exit program.
- The transmission queue must exist, be enabled for GETs, and have no other channels using it.
- MCAs, local and remote, must exist.
- The communication link must be available.
- The queue managers must be running, local and remote.
- The message channel must be inactive.

To transfer messages, remote queues and remote queue definitions must exist.

A message is returned to the panel confirming that the request to start a channel has been accepted. For confirmation that the Start process has succeeded, check the system log, or press F5 (refresh the screen).

15=End: IBM i

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.

To do restart the channel, issue one of the following commands:
- The START CHANNEL MQSC command
- The Start Channel PCF command
- the IBM MQ Explorer
- other platform-specific mechanisms, as follows:

  For z/OS:
  The Start a channel panel

  For IBM i:
  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 (on z/OS) or queue manager (on Multiplatforms) stops while a channel is in RETRYING or STOPPED status, the channel status is remembered when the channel initiator or queue manager is restarted. However, the channel status for the SVRCONN channel type is reset if the channel initiator or queue manager stops while the channel is in STOPPED status.

16=Reset: IBM i

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: IBM i

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: IBM i

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: IBM i

Use the Grant authority option to grant the authority to perform actions on IBM MQ objects to another user or group of users.

The GRTMQMAUT command is only available to users in the QMQMADM group. A user in QMQMADM grants authority to other users to perform actions on the IBM MQ objects named in the command either by identifying the users by name, or by granting authority to all users in *PUBLIC.

20=Revoke authority: IBM i

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: IBM i

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: IBM i

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

IBM i

When a distributed-queuing management channel is started, it tries to use the connection specified in the channel definition. For it to succeed, it is necessary for the connection to be defined and available.

DQM is a remote queuing facility for IBM MQ for IBM i. It provides channel control programs for the IBM MQ for IBM i queue manager which form the interface to communication links, controllable by the system operator.

When a distributed-queuing management channel is started, it tries to use the connection specified in the channel definition. For it to succeed, it is necessary for the connection to be defined and available. This section explains how to ensure that the connection is defined and available.

Before a channel can be started, the transmission queue must be defined as described in this section, and must be included in the message channel definition.

You can choose between the following two forms of communication between IBM MQ for IBM i systems:

- "Defining a TCP connection on IBM i"

  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 1056

  This form of communication requires the definition of an IBM i SNA logical unit type 6.2 (LU 6.2) that provides the physical link between the IBM i system serving the local queue manager and the system serving the remote queue manager. Refer to the *IBM i Communication Configuration Reference* for details on configuring communications in IBM i.

In addition, where needed, the triggering arrangement must be prepared with the definition of the necessary processes and queues.

Related tasks:

"Monitoring and controlling channels on IBM i" on page 1039
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 information:

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: 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.

Related concepts:

"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.

Receiving on TCP: IBM i

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 1000) 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: 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 121. Settings on the local IBM i system for a remote queue manager platform

Remote platform	**TPNAME**
z/OS or MVS	The same as in the corresponding side information about the remote queue manager.
IBM i	The same as the compare value in the routing entry on the IBM i system.
HP Integrity NonStop Server	The same as the TPNAME specified in the receiver-channel definition.
UNIX and Linux systems	The invokable Transaction Program defined in the remote LU 6.2 configuration.
Windows	As specified in the Windows Run Listener command, or the invokable Transaction Program that was defined using TpSetup on Windows.

If you have more than one queue manager on the same computer, ensure that the TPnames in the channel definitions are unique.

Related concepts:

"Initiating end (Sender)"
Use the CRTMQMCHL command to define a channel of transport type *LU62.

"Initiated end (Receiver)" on page 1061
Use the CRTMQMCHL command to define the receiving end of the message channel link with transport type *LU62.

Initiating end (Sender): IBM i

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 112. LU 6.2 communication setup panel - initiating end

Complete the initiating end fields as follows:

Side information
: Give this definition a name that is used to store the side information object to be created, for example, WINSDOA1.

  Note: For LU 6.2, the link between the message channel definition and the communication connection is the **Connection name** field of the message channel definition at the sending end. This field contains the name of the CSI object.

Library
: The name of the library where this definition is stored.

  The CSI object must be available in a library accessible to the program serving the message channel, for example, QSYS, QMQM, and QGPL.

  If the name is incorrect, missing, or cannot be found then an error occurs on channel startup.

Remote location
: Specifies the remote location name with which your program communicates.

In short, this required parameter contains the logical unit name of the partner at the remote system, as defined in the device description that is used for the communication link between the two systems.

The **Remote location** name can be found by issuing the command DSPNETA on the remote system and seeing the default local location name.

Transaction program

Specifies the name (up to 64 characters) of the transaction program on the remote system to be started. It can be a transaction process name, a program name, the channel name, or a character string that matches the **Compare value** in the routing entry.

This parameter is required.

Note: To specify SNA service transaction program names, enter the hexadecimal representation of the service transaction program name. For example, to specify a service transaction program name with a hexadecimal representation of 21F0F0F1, you would enter X'21F0F0F1'.

More information about SNA service transaction program names is in the *SNA Transaction Programmer's Reference* manual for LU Type 6.2.

If the receiving end is another IBM i system, the **Transaction program** name is used to match the CSI object at the sending end with the routing entry at the receiving end. This name must be unique for each queue manager on the target IBM i system. See the **Program to call** parameter under Initiated end (Receiver). See also the **Comparison data: compare value** parameter in the Add Routing Entry panel.

Text description

A description (up to 50 characters) to remind you of the intended use of this connection.

Device

Specifies the name of the device description used for the remote system. The possible values are:

***LOC** The device is determined by the system.

Device-name

Specify the name of the device that is associated with the remote location.

Local location

Specifies the local location name. The possible values are:

***LOC** The local location name is determined by the system.

***NETATR**

The LCLLOCNAME value specified in the system network attributes is used.

Local-location-name

Specify the name of your location. Specify the local location if you want to indicate a specific location name for the remote location. The location name can be found by using the DSPNETA command.

Mode Specifies the mode used to control the session. This name is the same as the Common Programming Interface (CPI)- Communications Mode_Name. The possible values are:

***NETATR**

The mode in the network attributes is used.

BLANK

Eight blank characters are used.

Mode-name

Specify a mode name for the remote location.

Note: Because the mode determines the transmission priority of the communications session, it might be useful to define different modes depending on the priority of the messages being sent; for example MQMODE_HI, MQMODE_MED, and MQMODE_LOW. (You can have more than one CSI pointing to the same location.)

Remote network identifier
Specifies the remote network identifier used with the remote location. The possible values are:

***LOC** The remote network ID for the remote location is used.

***NETATR**
The remote network identifier specified in the network attributes is used.

***NONE**
The remote network has no name.

Remote-network-id
Specify a remote network ID. Use the DSPNETA command at the remote location to find the name of this network ID. It is the 'local network ID' at the remote location.

Authority
Specifies the authority you are giving to users who do not have specific authority to the object, who are not on an authorization list, and with a group profile that has no specific authority to the object. The possible values are:

***LIBCRTAUT**
Public authority for the object is taken from the CRTAUT parameter of the specified library. This value is determined at create time. If the CRTAUT value for the library changes after the object is created, the new value does not affect existing objects.

***CHANGE**
Change authority allows the user to perform basic functions on the object, however, the user cannot change the object. Change authority provides object operational authority and all data authority.

***ALL** The user can perform all operations except those operations limited to the owner or controlled by authorization list management authority. The user can control the existence of the object and specify the security for the object, change the object, and perform basic functions on the object. The user can change ownership of the object.

***USE** Use authority provides object operational authority and read authority.

***EXCLUDE**
Exclude authority prevents the user from accessing the object.

Authorization-list
Specify the name of the authorization list with authority that is used for the side information.

Initiated end (Receiver):

Use the CRTMQMCHL command to define the receiving end of the message channel link with transport type *LU62.

Leave the CONNECTION NAME field blank and ensure that the corresponding details match the sending end of the channel. For details, see Creating a channel.

To enable the initiating end to start the receiving channel, add a routing entry to a subsystem at the initiated end. The subsystem must be the one that allocates the APPC device used in the LU 6.2 sessions. Therefore, it must have a valid communications entry for that device. The routing entry calls the program that starts the receiving end of the message channel.

Use the IBM i commands (for example, ADDRTGE) to define the end of the link that is initiated by a communication session.

The initiated end panel is shown in LU 6.2 communication setup panel - add routing entry.

```
Add Routing Entry (ADDRTGE)

Type choices, press Enter.

Subsystem description . . . . .  QCMN     Name
Library . . . . . . . . . . .   *LIBL    Name, *LIBL, *CURLIB
Routing entry sequence number .  1       1-9999
Comparison data:
Compare value . . . . . . . .  MQSERIES

Starting position . . . . . .  37      1-80
Program to call . . . . . . . .  AMQCRC6B   Name, *RTGDTA
Library . . . . . . . . . . .  QMAS400    Name, *LIBL, *CURLIB
Class . . . . . . . . . . . . .  *SBSD     Name, *SBSD
Library . . . . . . . . . . .  *LIBL     Name, *LIBL, *CURLIB
Maximum active routing steps . .  *NOMAX    0-1000, *NOMAX
Storage pool identifier . . . .  1       1-10



Bottom
F3=Exit  F4=Prompt  F5=Refresh  F12=Cancel  F13=How to use this display
F24=More keys
```

Figure 113. LU 6.2 communication setup panel - initiated end

Subsystem description
: The name of your subsystem where this definition resides. Use the IBM i WRKSBSD command to view and update the appropriate subsystem description for the routing entry.

Routing entry sequence number
: A unique number in your subsystem to identify this communication definition. You can use values in the range 1 - 9999.

Comparison data: Compare value
: A text string to compare with the string received when the session is started by a **Transaction program** parameter, as shown in Figure 1. The character string is derived from the Transaction program field of the sender CSI.

Comparison data: Starting position
: The character position in the string where the comparison is to start.

  Note: The starting position field is the character position in the string for comparison, and this position is always 37.

Program to call

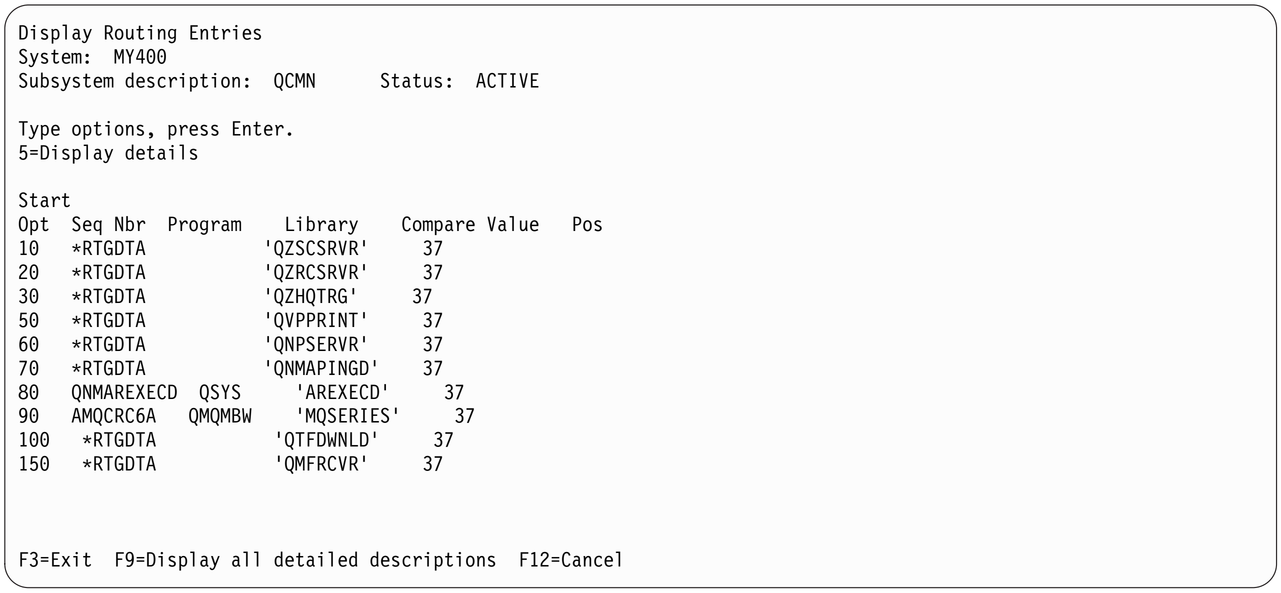
The name of the program that runs the inbound message program to be called to start the session.

The program, AMQCRC6A, is called for the default queue manager. This program is supplied with IBM MQ for IBM i and sets up the environment and then calls AMQCRS6A.

For additional queue managers:

- Each queue manager has a specific LU 6.2 invokable program located in its library. This program is called AMQCRC6B and is automatically generated when the queue manager is created.
- Each queue manager requires a specific routing entry with unique routing data to be added. This routing data must match the **Transaction program** name supplied by the requesting system (see Initiating end (Sender)).

An example is shown in LU 6.2 communication setup panel - display routing entries:



```
Display Routing Entries
System:  MY400
Subsystem description:  QCMN      Status:  ACTIVE

Type options, press Enter.
5=Display details

Start
Opt  Seq Nbr  Program    Library    Compare Value   Pos
10   *RTGDTA           'QZSCSRVR'     37
20   *RTGDTA           'QZRCSRVR'     37
30   *RTGDTA           'QZHQTRG'     37
50   *RTGDTA           'QVPPRINT'     37
60   *RTGDTA           'QNPSERVR'     37
70   *RTGDTA           'QNMAPINGD'    37
80   QNMAREXECD  QSYS     'AREXECD'     37
90   AMQCRC6A   QMQMBW    'MQSERIES'     37
100   *RTGDTA           'QTFDWNLD'     37
150   *RTGDTA           'QMFRCVR'     37



F3=Exit  F9=Display all detailed descriptions  F12=Cancel
```

Figure 114. 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 1172
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 information:

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:

"Setting up a new cluster" on page 1074
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 1085
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`.

Related information:

Defining cluster topics

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 MQOO_BIND_ON_OPEN option on the MQOPEN call.

Cluster transmission queues

A queue manager can store messages for other queue managers in a cluster on multiple transmission queues. You can configure a queue manager to store messages on multiple cluster transmission queues in two different ways. If you set the queue manager attribute DEFCLXQ to CHANNEL, a different cluster transmission queue is created automatically from SYSTEM.CLUSTER.TRANSMIT.MODEL.QUEUE for each cluster-sender channel. If you set the CLCHNAME transmission queue option to match one or more cluster-senders channel, the queue manager can store messages for the matching channels on that transmission queue.

A message for a cluster queue on a different queue manager is placed upon a cluster transmission queue before being sent. A cluster-sender channel transfers the messages from a cluster transmission queue to cluster-receiver channels on other queue managers. By default, one system defined cluster transmission queue holds all the messages that are to be transferred to other cluster queue managers. The queue is called SYSTEM.CLUSTER.TRANSMIT.QUEUE. A queue manager that is part of a cluster can send messages on this cluster transmission queue to any other queue manager in the same cluster.

A definition for the single SYSTEM.CLUSTER.TRANSMIT.QUEUE queue is created by default on every queue manager except on z/OS. z/OS On z/OS, the definition can be defined with the supplied sample **CSQ4INSX**.

You can configure a queue manager to transfer messages to other clustered queue managers using multiple transmission queues. You can define additional cluster transmission queues manually, or have the queue manager create the queues automatically.

To have the queues created automatically by the queue manager, change the queue manager attribute DEFCLXQ from SCTQ to CHANNEL. The result is the queue manager creates an individual cluster transmission queue for each cluster-sender channel that is created. The transmission queues are created as permanent dynamic queues from the model queue, SYSTEM.CLUSTER.TRANSMIT.MODEL.QUEUE. The name of each permanent dynamic queue is SYSTEM.CLUSTER.TRANSMIT. *ChannelName*. The name of the cluster-sender channel that each permanent dynamic cluster transmit queue is associated with is set in the local transmission queue attribute CLCHNAME. Messages for remote clustered queue managers are placed on the permanent dynamic cluster transmission queue for the associated cluster-sender channel, rather than on SYSTEM.CLUSTER.TRANSMIT.QUEUE.

To create the cluster transmission queues manually, create a local queue with the USAGE attribute set to XMITQ, and the CLCHNAME attribute set to a generic channel name that resolves to one or more cluster-sender channels; see ClusterChannelName. If you create cluster transmission queues manually, you have the choice of associating the transmission queue with a single cluster-sender channel, or with multiple cluster-sender channels. The CLCHNAME attribute is a generic-name, which means you can place multiple wildcard characters, "*", in the name.

Except for the initial cluster-sender channels that you create manually to connect a queue manager to a full repository, cluster-sender channels are created automatically. They are created automatically when there is a message to transfer to a cluster queue manager. They are created with the same name as the name of the cluster-receiver channel that receives cluster messages for that particular cluster on the destination queue manager.

If you follow a naming convention for cluster-receiver channels, it is possible to define a generic value for CLCHNAME that filters different kinds of cluster messages to different transmission queues. For example, if you follow the naming convention for cluster-receiver channels of *ClusterName*. *QmgrName*, then the generic name *ClusterName*.* filters messages for different clusters onto different transmission queues. You must define the transmission queues manually, and set CLCHNAME in each transmission queue to *ClusterName*.*.

Changes to the association of cluster transmission queues to cluster-sender channels do not take immediate effect. The currently associated transmission queue that a cluster-sender channel is servicing might contain messages that are in the process of being transferred by the cluster-sender channel. Only when no messages on the currently associated transmission queue are being processed by a cluster-sender channel, can the queue manager change the association of the cluster-sender channel a different transmission queue. This can occur either when no messages remain on the transmission queue to be processed by the cluster-sender channel, or when processing of messages is suspended and the cluster-sender channel has no "in-flight" messages. When this happens, any unprocessed messages for the cluster-sender channel are transferred to the newly associated transmission queue, and the association of the cluster-sender channel changes.

You can create a remote queue definition that resolves to a cluster transmission queue. In the definition, queue manager QMX is in the same cluster as the local queue manager, and there is no transmission queue, QMX.

```
DEFINE QREMOTE(A) RNAME(B) RQMNAME(QMX)
```

During queue name resolution, the cluster transmission queue takes precedence over the default transmission queue. A message put to A is stored on the cluster transmission queue and then sent to the remote queue B on QMX.

Queue managers can also communicate with other queue managers that are not part of a cluster. You must define channels and a transmission queue to the other queue manager, in the same way as in a distributed-queuing environment.

Note: Applications must write to queues that resolve to the cluster transmission queue, and must not write directly to the cluster transmission queue.

Auto definition of remote queues

A queue manager in a cluster does not need a remote-queue definition for remote queues in the cluster. The cluster queue manager finds the location of a remote queue from the full repository. It adds routing information to the message and puts it on the cluster transmission queue. IBM MQ automatically creates a definition equivalent to a remote-queue definition so that the message can be sent.

You cannot alter or delete an automatically created remote-queue definition. However, by using the `DISPLAY QUEUE` **runmqsc** command with the `CLUSINFO` attribute, you can view all of the local queues on a queue manager as well as all of the cluster queues, including cluster queues on remote queue managers. For example:

```
DISPLAY QUEUE(*) CLUSINFO
```

Related information:

Cluster queues

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 they are needed. 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.

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` **runmqsc** command. To see the auto-defined channels use the following command:

  ```
  DISPLAY CLUSQMGR(qMgrName)
  ```
- Display the status of the auto-defined channel for a given `CLUSRCVR`.

  To display the status of the auto-defined `CLUSSDR` channel corresponding to a `CLUSRCVR` channel definition you created, use the following command:

  ```
  DISPLAY CHSTATUS(channelname)
  ```
- Use a channel auto-definition exit to modify the behavior of an auto-defined channel.

  You can use the IBM MQ channel auto-definition exit if you want to write a user exit program to customize a cluster-sender channel or cluster-receiver channel. For example, you can use the channel auto-definition exit in a cluster environment to make any of the following modifications:

- Tailor communications definitions, that is, SNA LU6.2 names.
- Add or remove other exits, for example, security exits.
- Change the names of channel exits.

  The name of the CLUSSDR channel exit is auto-generated from the CLUSRCVR channel definition, and therefore might not be appropriate for your needs - especially if the two ends of the channel are on different platforms.

  The format of exit names is different on different platforms. For example:

  - z/OS On the z/OS platform, the format of the SCYEXIT (*security exit name*) parameter is SCYEXIT('SECEXIT')
  - Windows On Windows platforms, the format of the SCYEXIT (*security exit name*) parameter is SCYEXIT(' *drive*:\path\library (secexit)')

  Note: z/OS If there is no channel auto-definition exit, the z/OS queue manager derives the CLUSSDR channel exit name from the CLUSRCVR channel definition on the other end of the channel. To derive the z/OS exit name from a non-z/OS name, the following algorithm is used:

  - Exit names on 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
- For queue managers earlier than IBM MQ Version 7, set the **PROPCTL** attribute to a value of NONE.

  Each auto-defined cluster-sender channel is based on the corresponding cluster-receiver channel. Before IBM MQ Version 7, the cluster-receiver channel does not have a **PROPCTL** attribute, so this attribute is therefore set to COMPAT in the auto-defined cluster-sender channel.

  If the cluster needs to use **PROPCTL** to remove application headers such as RFH2 from messages going from an IBM MQ Version 7 or later queue manager to a queue manager on an earlier version of IBM MQ, you must write a channel auto-definition exit that sets **PROPCTL** to a value of NONE.

- Use the channel attribute LOCLADDR to control aspects of addressing.
  - To enable an outbound (TCP) channel to use a particular IP address, port or port range, use the channel attribute LOCLADDR. This is useful if you have more than one network card and you want a channel to use a specific one for outbound communications.
  - To specify a virtual IP address on CLUSSDR channels, use the IP address from the LOCLADDR on a manually defined CLUSSDR. To specify the port range, use the port range from the CLUSRCVR.
  - If a cluster needs to use LOCLADDR to get the outbound communication channels to bind to a specific IP address, you can write a channel auto-definition exit to force the LOCLADDR value into any of their automatically defined CLUSSDR channels. You must also specify it in the manually defined CLUSSDR channel.
  - Put a port number or port range in the LOCLADDR of a CLUSRCVR channel, if you want all the queue managers in a cluster to use a specific port or range of ports, for all their outbound communications.

Note: Do not put an IP address in the LOCLADDR field of a CLUSRCVR channel, unless all queue managers are on the same server. The LOCLADDR IP address is propagated to the auto-defined CLUSSDR channels of all queue managers that connect using the CLUSRCVR channel.

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 information:
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`. `SYSTEM.CLUSTER.HISTORY.QUEUE` is used to store the history of cluster state information for service purposes.

  In the default object settings, `SYSTEM.CLUSTER.HISTORY.QUEUE` is set to `PUT ( ENABLED)`. To suppress history collection change the setting to `PUT ( DISABLED)`.

`SYSTEM.CLUSTER.TRANSMIT.QUEUE`
: Each queue manager has a definition for a local queue called `SYSTEM.CLUSTER.TRANSMIT.QUEUE`. `SYSTEM.CLUSTER.TRANSMIT.QUEUE` is the default transmission queue for all messages to all queues and queue managers that are within clusters. You can change the default transmission queue for each cluster-sender channel to `SYSTEM.CLUSTER.TRANSMIT.` *`ChannelName`*, by changing the queue manager attribute `DEFXMITQ` z/OS , except on z/OS . You cannot delete `SYSTEM.CLUSTER.TRANSMIT.QUEUE`. It is also used to define authorization checks whether the default transmission queue that is used is `SYSTEM.CLUSTER.TRANSMIT.QUEUE` or `SYSTEM.CLUSTER.TRANSMIT.` *`ChannelName`*.

Related information:
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 1069 describes the process of selecting a default transmission queue; “Selection of manually defined transmission queues by cluster-sender channels” on page 1070 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 1071 describes the process of transferring a cluster-sender channel from one transmission queue to another.

The behavior of cluster-sender channels is different to sender and server channels. They remain associated with the same transmission queue until the channel attribute **XMITQ** is altered. If you alter the transmission queue attribute on a sender or server channel, and restart it, messages are not transferred from the old transmission queue to the new one.

Another difference between cluster-sender channels, and sender or server channels, is that multiple cluster-sender channels can open a cluster transmission queue, but only one sender or server channel can open a normal transmission queue. Until Version 7.5, cluster connections shared the single cluster transmission queue, SYSTEM.CLUSTER.TRANSMIT.QUEUE. From Version 7.5 onwards, you have the option of cluster-sender channels not sharing transmission queues. Exclusivity is not enforced; it is an outcome of the configuration. You can configure the path a message takes in a cluster so that it does not share any transmission queues or channels with messages that flow between other applications. See Clustering: Planning how to configure cluster transmission queues and "Adding a cluster and a cluster transmit queue to isolate cluster message traffic sent from a gateway queue manager" on page 1118.

To configure a cluster-sender channel to use a transmission queue other than SYSTEM.CLUSTER.TRANSMIT.QUEUE on z/OS, you need to enable version 8 new function, using the mode of operation (OPMODE) system parameter in the CSQ6SYSP macro.

Selection of default transmission queues by cluster-sender channels

A cluster transmission queue is either a system default queue, with a name that starts with SYSTEM.CLUSTER.TRANSMIT, or a manually defined queue. A cluster-sender channel is associated with a cluster transmission queue in one of two ways: by the default cluster transmission queue mechanism, or by manual configuration.

The default cluster transmission queue is set as a queue manager attribute, **DEFCLXQ**. Its value is either SCTQ or CHANNEL. New and migrated queue managers are set to SCTQ. You can alter the value to CHANNEL.

If SCTQ is set, the default cluster transmission queue is SYSTEM.CLUSTER.TRANSMIT.QUEUE. Every cluster-sender channel can open this queue. The cluster-sender channels that do open the queue are the ones that are not associated with manually defined cluster transmission queues.

If CHANNEL is set, then the queue manager can create a separate permanent dynamic transmission queue for every cluster-sender channel. Each queue is named SYSTEM.CLUSTER.TRANSMIT. *ChannelName* and is created from the model queue, SYSTEM.CLUSTER.TRANSMIT.MODEL.QUEUE. Each cluster-sender channel that is not associated with a manually defined cluster transmission queue is associated with a permanent-dynamic cluster transmission queue. The queue is created by the queue manager when it requires a separate cluster transmission queue for the cluster destination served by this cluster-sender channel, and no queue exists.

Some cluster destinations can be served by cluster-sender channels associated with manually defined transmission queues, and others by the default queue or queues. In the association of cluster-sender channels with transmission queues, the manually defined transmission queues always take precedence over the default transmission queues.

The precedence of cluster transmission queues is illustrated in Figure 115 on page 1070. 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.

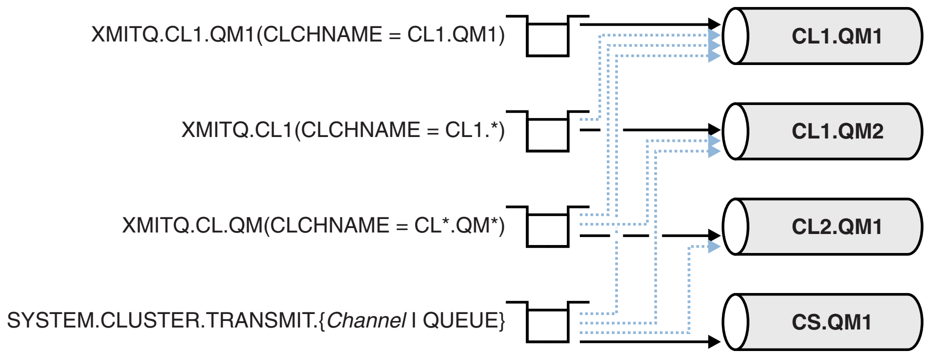

XMITQ.CL1.QM1(CLCHNAME = CL1.QM1)
XMITQ.CL1(CLCHNAME = CL1.*)
XMITQ.CL.QM(CLCHNAME = CL*.QM*)
SYSTEM.CLUSTER.TRANSMIT.{Channel | QUEUE}
CL1.QM1
CL1.QM2
CL2.QM1
CS.QM1

Figure 115. 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 115 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 115 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;l 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 116 on page 1073, `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 116, 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.

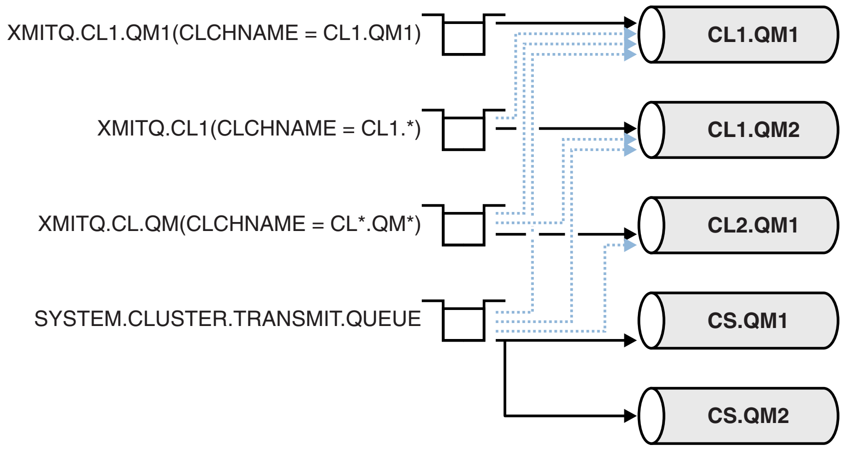

XMITQ.CL1.QM1(CLCHNAME = CL1.QM1)
XMITQ.CL1(CLCHNAME = CL1.*)
XMITQ.CL.QM(CLCHNAME = CL*.QM*)
SYSTEM.CLUSTER.TRANSMIT.QUEUE
CL1.QM1
CL1.QM2
CL2.QM1
CS.QM1
CS.QM2

Figure 116. 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 1071.

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:

"Calculating the size of the log" on page 1331
Estimating the size of log a queue manager needs.

Related tasks:

"Creating two-overlapping clusters with a gateway queue manager" on page 1108
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 1087
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 1115
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 1118
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.

Related information:

Cluster channels

Clustering: Application isolation using multiple cluster transmission queues

Clustering: Planning how to configure cluster transmission queues

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 1064, Cluster channels and Listeners.

About this task

You are setting up a new IBM MQ network for a chain store. The store has two branches, one in London and one in New York. The data and applications for each store are hosted by systems running separate queue managers. The two queue managers are called `LONDON` and `NEWYORK`. The inventory application runs on the system in New York, connected to queue manager `NEWYORK`. The application is driven by the arrival of messages on the `INVENTQ` queue, hosted by `NEWYORK`. The two queue managers, `LONDON` and `NEWYORK`, are to be linked in a cluster called `INVENTORY` so that they can both put messages to the `INVENTQ`.

Figure 117 shows what this cluster looks like.

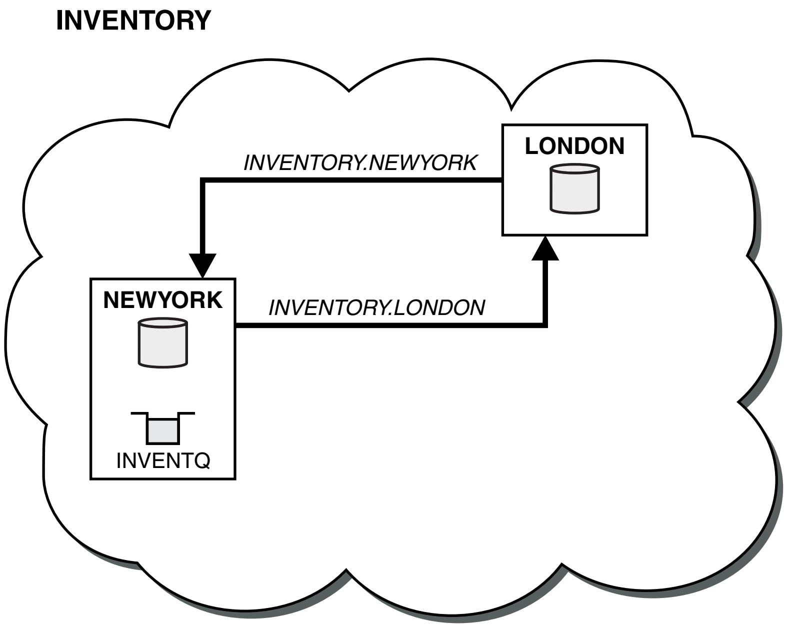


Figure 117. The INVENTORY cluster with two queue managers

You can configure each queue manager in the cluster to send messages to other queue managers in the cluster using different cluster transmission queues.

The instructions to set up the cluster vary a little by transport protocol , number of transmission queues, or platform. You have a choice of three combinations. The verification procedure remains the same for all combinations.

Procedure

- "Setting up a cluster using TCP/IP with a single transmission queue per queue manager" on page 1076
- "Setting up a cluster on TCP/IP using multiple transmission queues per queue manager" on page 1078
- "Setting up a cluster using LU 6.2 on z/OS" on page 1081
- "Verifying the cluster" on page 1083

Results

Figure 117 on page 1075 shows the INVENTORY cluster set up by this task.

Clearly, INVENTORY is a small cluster. However, it is useful as a proof of concept. The important thing to understand about this cluster is the scope it offers for future enhancement.

Related tasks:

"Configuring a queue manager cluster" on page 1063
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.

Related information:

Clusters

Comparison of clustering and distributed queuing

Components of a cluster

Setting up a cluster using TCP/IP with a single transmission queue per queue manager:

Before you begin

- The queue manager attribute, **DEFCLXQ**, must be left as its default value, SCTQ.

About this task

Follow these steps to set up a cluster on 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 1554 to set up the TCP/IP connection , rather than defining the listeners in step 4 on page 1077. 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 1085.

   For the time being, the only application you are running is the inventory application. The cluster name is INVENTORY.
2. Decide which queue managers are to hold full repositories.

   In any cluster you must nominate at least one queue manager, or preferably two, to hold full repositories. In this example, there are only two queue managers, LONDON and NEWYORK, both of which hold full repositories.
   a. You can perform the remaining steps in any order.
   b. As you proceed through the steps, warning messages might be written to the queue manager log. The messages are a result of missing definitions that you have yet to add.

```
Examples of the responses to the commands are shown in a box
like this after each step in this task.
These examples show the responses returned by IBM MQ for AIX.
The responses vary on other platforms.
```

   c. Before proceeding with these steps, make sure that the queue managers are started.

3. Alter the queue manager definitions to add repository definitions.

   On each queue manager that is to hold a full repository, alter the local queue manager definition, using the `ALTER QMGR` command and specifying the `REPOS` attribute:

   ```
   ALTER QMGR REPOS(INVENTORY)
   ```

   ```
   1 : ALTER QMGR REPOS(INVENTORY)
   AMQ8005: WebSphere MQ queue manager changed.
   ```

   For example, if you enter:

   a. `runmqsc LONDON`
   b. `ALTER QMGR REPOS(INVENTORY)`

   `LONDON` is changed to a full repository.

4. Define the listeners.

   Define a listener that accepts network requests from other queue managers for every queue manager in the cluster. On the `LONDON` queue managers, issue the following command:

   ```
   DEFINE LISTENER(LONDON_LS) TRPTYPE(TCP) CONTROL(QMGR)
   ```

   The `CONTROL` attribute ensures that the listener starts and stops when the queue manager does.

   The listener is not started when it is defined, so it must be manually started the first time with the following MQSC command:

   ```
   START LISTENER(LONDON_LS)
   ```

   Issue similar commands for all the other queue managers in the cluster, changing the listener name for each one.

   There are several ways to define these listeners, as shown in Listeners.

5. Define the CLUSRCVR channel for the `LONDON` queue manager.

   On every queue manager in a cluster, you define a cluster-receiver channel on which the queue manager can receive messages. See Cluster-receiver channel: `CLUSRCVR` . The `CLUSRCVR` channel defines the connection name of the queue manager. The connection name is stored in the repositories, where other queue managers can refer to it. The `CLUSTER` keyword shows the availability of the queue manager to receive messages from other queue managers in the cluster.

   In this example the channel name is `INVENTORY.LONDON`, and the connection name (`CONNAME`) is the network address of the machine the queue manager resides on, which is `LONDON.CHSTORE.COM`. The network address can be entered as an alphanumeric DNS host name, or an IP address in either in IPv4 dotted decimal form. For example, `192.0.2.0`, or IPv6 hexadecimal form; for example `2001:DB8:0204:acff:fe97:2c34:fde0:3485`. The port number is not specified, so the default port (1414) is used.

   ```
   DEFINE CHANNEL(INVENTORY.LONDON) CHLTYPE(CLUSRCVR) TRPTYPE(TCP)
   CONNAME(LONDON.CHSTORE.COM) CLUSTER(INVENTORY)
   DESCR('TCP Cluster-receiver channel for queue manager LONDON')
   ```

```
1 : DEFINE CHANNEL(INVENTORY.LONDON) CHLTYPE(CLUSRCVR) TRPTYPE(TCP)
CONNAME(LONDON.CHSTORE.COM) CLUSTER(INVENTORY)
DESCR('TCP Cluster-receiver channel for queue manager LONDON')
AMQ8014: WebSphere MQ channel created.
07/09/98 12:56:35 No repositories for cluster 'INVENTORY'
```

6. Define the CLUSRCVR channel for the NEWYORK queue manager. If the channel listener is using the default port, typically 1414, and the cluster does not include a queue manager on z/OS, you can omit the CONNAME

   ```
   DEFINE CHANNEL(INVENTORY.NEWYORK) CHLTYPE(CLUSRCVR) TRPTYPE(TCP) CLUSTER(INVENTORY)
   DESCR('TCP Cluster-receiver channel for queue manager NEWYORK')
   ```

7. Define the CLUSSDR channel on the LONDON queue manager.

   You manually define a CLUSSDR channel from every full repository queue manager to every other full repository queue manager in the cluster. See Cluster-sender channel: CLUSSDR . In this case, there are only two queue managers, both of which hold full repositories. They each need a manually-defined CLUSSDR channel that points to the CLUSRCVR channel defined at the other queue manager. The channel names given on the CLUSSDR definitions must match the channel names on the corresponding CLUSRCVR definitions. When a queue manager has definitions for both a cluster-receiver channel and a cluster-sender channel in the same cluster, the cluster-sender channel is started.

   ```
   DEFINE CHANNEL(INVENTORY.NEWYORK) CHLTYPE(CLUSSDR) TRPTYPE(TCP)
   CONNAME(NEWYORK.CHSTORE.COM) CLUSTER(INVENTORY)
   DESCR('TCP Cluster-sender channel from LONDON to repository at NEWYORK')
   ```

```
1 : DEFINE CHANNEL(INVENTORY.NEWYORK) CHLTYPE(CLUSSDR) TRPTYPE(TCP)
CONNAME(NEWYORK.CHSTORE.COM) CLUSTER(INVENTORY)
DESCR('TCP Cluster-sender channel from LONDON to repository at NEWYORK')
AMQ8014: WebSphere MQ channel created.
07/09/98 13:00:18  Channel program started.
```

8. Define the CLUSSDR channel on the NEWYORK queue manager.

   ```
   DEFINE CHANNEL(INVENTORY.LONDON) CHLTYPE(CLUSSDR) TRPTYPE(TCP)
   CONNAME(LONDON.CHSTORE.COM) CLUSTER(INVENTORY)
   DESCR('TCP Cluster-sender channel from NEWYORK to repository at LONDON')
   ```

9. Define the cluster queue INVENTQ

   Define the INVENTQ queue on the NEWYORK queue manager, specifying the CLUSTER keyword.

   ```
   DEFINE QLOCAL(INVENTQ) CLUSTER(INVENTORY)
   ```

```
1 : DEFINE QLOCAL(INVENTQ) CLUSTER(INVENTORY)
AMQ8006: WebSphere MQ queue created.
```

The CLUSTER keyword causes the queue to be advertised to the cluster. As soon as the queue is defined it becomes available to the other queue managers in the cluster. They can send messages to it without having to make a remote-queue definition for it.

All the definitions are complete. On all platforms, start a listener program on each queue manager. The listener program waits for incoming network requests and starts the cluster-receiver channel when it is needed.

Setting up a cluster on TCP/IP using multiple transmission queues per queue manager:

About this task

Follow these steps to set up a cluster on 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 1087.

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 1085.

   For the time being, the only application you are running is the inventory application. The cluster name is `INVENTORY`.
2. Decide which queue managers are to hold full repositories.

   In any cluster you must nominate at least one queue manager, or preferably two, to hold full repositories. In this example, there are only two queue managers, `LONDON` and `NEWYORK`, both of which hold full repositories.
   a. You can perform the remaining steps in any order.
   b. As you proceed through the steps, warning messages might be written to the queue manager log. The messages are a result of missing definitions that you have yet to add.

```
Examples of the responses to the commands are shown in a box
like this after each step in this task.
These examples show the responses returned by IBM MQ for AIX.
The responses vary on other platforms.
```

   c. Before proceeding with these steps, make sure that the queue managers are started.
3. Alter the queue manager definitions to add repository definitions.

   On each queue manager that is to hold a full repository, alter the local queue manager definition, using the `ALTER QMGR` command and specifying the `REPOS` attribute:

   ```
   ALTER QMGR REPOS(INVENTORY)
   ```

```
1 : ALTER QMGR REPOS(INVENTORY)
AMQ8005: WebSphere MQ queue manager changed.
```

   For example, if you enter:
   a. `runmqsc LONDON`
   b. `ALTER QMGR REPOS(INVENTORY)`

   `LONDON` is changed to a full repository.
4. Alter the queue manager definitions to create separate cluster transmission queues for each destination.

   ```
   ALTER QMGR DEFCLXQ(CHANNEL)
   ```

On each queue manager that you add to the cluster decide whether to use separate transmission queues or not. See the topics, “Adding a queue manager to a cluster” on page 1085 and “Adding a queue manager to a cluster: separate transmission queues” on page 1087.

5. Define the listeners.

   Define a listener that accepts network requests from other queue managers for every queue manager in the cluster. On the LONDON queue managers, issue the following command:

   ```
   DEFINE LISTENER(LONDON_LS) TRPTYPE(TCP) CONTROL(QMGR)
   ```

   The CONTROL attribute ensures that the listener starts and stops when the queue manager does.

   The listener is not started when it is defined, so it must be manually started the first time with the following MQSC command:

   ```
   START LISTENER(LONDON_LS)
   ```

   Issue similar commands for all the other queue managers in the cluster, changing the listener name for each one.

   There are several ways to define these listeners, as shown in Listeners.

6. Define the CLUSRCVR channel for the LONDON queue manager.

   On every queue manager in a cluster, you define a cluster-receiver channel on which the queue manager can receive messages. See Cluster-receiver channel: CLUSRCVR . The CLUSRCVR channel defines the connection name of the queue manager. The connection name is stored in the repositories, where other queue managers can refer to it. The CLUSTER keyword shows the availability of the queue manager to receive messages from other queue managers in the cluster.

   In this example the channel name is INVENTORY.LONDON, and the connection name (CONNAME) is the network address of the machine the queue manager resides on, which is LONDON.CHSTORE.COM. The network address can be entered as an alphanumeric DNS host name, or an IP address in either in IPv4 dotted decimal form. For example, 192.0.2.0, or IPv6 hexadecimal form; for example 2001:DB8:0204:acff:fe97:2c34:fde0:3485. The port number is not specified, so the default port (1414) is used.

   ```
   DEFINE CHANNEL(INVENTORY.LONDON) CHLTYPE(CLUSRCVR) TRPTYPE(TCP)
   CONNAME(LONDON.CHSTORE.COM) CLUSTER(INVENTORY)
   DESCR('TCP Cluster-receiver channel for queue manager LONDON')
   ```

```
1 : DEFINE CHANNEL(INVENTORY.LONDON) CHLTYPE(CLUSRCVR) TRPTYPE(TCP)
CONNAME(LONDON.CHSTORE.COM) CLUSTER(INVENTORY)
DESCR('TCP Cluster-receiver channel for queue manager LONDON')
AMQ8014: WebSphere MQ channel created.
07/09/98 12:56:35 No repositories for cluster 'INVENTORY'
```

7. Define the CLUSRCVR channel for the NEWYORK queue manager. If the channel listener is using the default port, typically 1414, and the cluster does not include a queue manager on z/OS, you can omit the CONNAME

   ```
   DEFINE CHANNEL(INVENTORY.NEWYORK) CHLTYPE(CLUSRCVR) TRPTYPE(TCP) CLUSTER(INVENTORY)
   DESCR('TCP Cluster-receiver channel for queue manager NEWYORK')
   ```

8. Define the CLUSSDR channel on the LONDON queue manager.

   You manually define a CLUSSDR channel from every full repository queue manager to every other full repository queue manager in the cluster. See Cluster-sender channel: CLUSSDR . In this case, there are only two queue managers, both of which hold full repositories. They each need a manually-defined CLUSSDR channel that points to the CLUSRCVR channel defined at the other queue manager. The channel names given on the CLUSSDR definitions must match the channel names on the corresponding CLUSRCVR definitions. When a queue manager has definitions for both a cluster-receiver channel and a cluster-sender channel in the same cluster, the cluster-sender channel is started.

```
DEFINE CHANNEL(INVENTORY.NEWYORK) CHLTYPE(CLUSSDR) TRPTYPE(TCP)
CONNAME(NEWYORK.CHSTORE.COM) CLUSTER(INVENTORY)
DESCR('TCP Cluster-sender channel from LONDON to repository at NEWYORK')
```

```
1 : DEFINE CHANNEL(INVENTORY.NEWYORK) CHLTYPE(CLUSSDR) TRPTYPE(TCP)
CONNAME(NEWYORK.CHSTORE.COM) CLUSTER(INVENTORY)
DESCR('TCP Cluster-sender channel from LONDON to repository at NEWYORK')
AMQ8014: WebSphere MQ channel created.
07/09/98 13:00:18  Channel program started.
```

9. Define the CLUSSDR channel on the NEWYORK queue manager.

   ```
   DEFINE CHANNEL(INVENTORY.LONDON) CHLTYPE(CLUSSDR) TRPTYPE(TCP)
   CONNAME(LONDON.CHSTORE.COM) CLUSTER(INVENTORY)
   DESCR('TCP Cluster-sender channel from NEWYORK to repository at LONDON')
   ```
10. Define the cluster queue INVENTQ

    Define the INVENTQ queue on the NEWYORK queue manager, specifying the CLUSTER keyword.

    ```
    DEFINE QLOCAL(INVENTQ) CLUSTER(INVENTORY)
    ```

```
1 : DEFINE QLOCAL(INVENTQ) CLUSTER(INVENTORY)
AMQ8006: WebSphere MQ queue created.
```

The CLUSTER keyword causes the queue to be advertised to the cluster. As soon as the queue is defined it becomes available to the other queue managers in the cluster. They can send messages to it without having to make a remote-queue definition for it.

All the definitions are complete. On all platforms, start a listener program on each queue manager. The listener program waits for incoming network requests and starts the cluster-receiver channel when it is needed.

Setting up a cluster using LU 6.2 on z/OS:
Procedure

1. Decide on the organization of the cluster and its name.

   You decided to link the two queue managers, LONDON and NEWYORK, into a cluster. A cluster with only two queue managers offers only marginal benefit over a network that is to use distributed queuing. It is a good way to start and it provides scope for future expansion. When you open new branches of your store, you are able to add the new queue managers to the cluster easily. Adding new queue managers does not disrupt the existing network; see “Adding a queue manager to a cluster” on page 1085.

   For the time being, the only application you are running is the inventory application. The cluster name is INVENTORY.
2. Decide which queue managers are to hold full repositories.

   In any cluster you must nominate at least one queue manager, or preferably two, to hold full repositories. In this example, there are only two queue managers, LONDON and NEWYORK, both of which hold full repositories.
   a. You can perform the remaining steps in any order.
   b. As you proceed through the steps, warning messages might be written the z/OS system console. The messages are a result of missing definitions that you have yet to add.
   c. Before proceeding with these steps, make sure that the queue managers are started.
3. Alter the queue manager definitions to add repository definitions.

On each queue manager that is to hold a full repository, alter the local queue manager definition, using the ALTER QMGR command and specifying the REPOS attribute:

```
ALTER QMGR REPOS(INVENTORY)
```

```
1 : ALTER QMGR REPOS(INVENTORY)
AMQ8005: WebSphere MQ queue manager changed.
```

For example, if you enter:

a. runmqsc LONDON
b. ALTER QMGR REPOS(INVENTORY)

LONDON is changed to a full repository.

4. Define the listeners.

   z/OS See The channel initiator on z/OS and “Receiving on LU 6.2” on page 1558.

   The listener is not started when it is defined, so it must be manually started the first time with the following MQSC command:

   ```
   START LISTENER(LONDON_LS)
   ```

   Issue similar commands for all the other queue managers in the cluster, changing the listener name for each one.

5. Define the CLUSRCVR channel for the LONDON queue manager.

   On every queue manager in a cluster, you define a cluster-receiver channel on which the queue manager can receive messages. See Cluster-receiver channel: CLUSRCVR . The CLUSRCVR channel defines the connection name of the queue manager. The connection name is stored in the repositories, where other queue managers can refer to it. The CLUSTER keyword shows the availability of the queue manager to receive messages from other queue managers in the cluster.

   ```
   DEFINE CHANNEL(INVENTORY.LONDON) CHLTYPE(CLUSRCVR) TRPTYPE(LU62)
   CONNAME(LONDON.LUNAME) CLUSTER(INVENTORY)
   MODENAME('#INTER') TPNAME('MQSERIES')
   DESCR('LU62 Cluster-receiver channel for queue manager LONDON')
   ```

```
1 : DEFINE CHANNEL(INVENTORY.LONDON) CHLTYPE(CLUSRCVR) TRPTYPE(LU62)
CONNAME(LONDON.LUNAME) CLUSTER(INVENTORY)
MODENAME('#INTER') TPNAME('MQSERIES')
DESCR('LU62 Cluster-receiver channel for queue manager LONDON')
AMQ8014: WebSphere MQ channel created.
07/09/98 12:56:35 No repositories for cluster 'INVENTORY'
```

6. Define the CLUSRCVR channel for the NEWYORK queue manager.

   ```
   DEFINE CHANNEL(INVENTORY.NEWYORK) CHLTYPE(CLUSRCVR) TRPTYPE(LU62)
   CONNAME(NEWYORK.LUNAME) CLUSTER(INVENTORY)
   MODENAME('#INTER') TPNAME('MQSERIES')
   DESCR('LU62 Cluster-receiver channel for queue manager NEWYORK')
   ```

7. Define the CLUSSDR channel on the LONDON queue manager.

   You manually define a CLUSSDR channel from every full repository queue manager to every other full repository queue manager in the cluster. See Cluster-sender channel: CLUSSDR . In this case, there are only two queue managers, both of which hold full repositories. They each need a manually-defined CLUSSDR channel that points to the CLUSRCVR channel defined at the other queue manager. The channel names given on the CLUSSDR definitions must match the channel names on the corresponding CLUSRCVR definitions. When a queue manager has definitions for both a cluster-receiver channel and a cluster-sender channel in the same cluster, the cluster-sender channel is started.

```
DEFINE CHANNEL(INVENTORY.NEWYORK) CHLTYPE(CLUSSDR) TRPTYPE(LU62)
CONNAME(CPIC) CLUSTER(INVENTORY)
DESCR('LU62 Cluster-sender channel from LONDON to repository at NEWYORK')
```

```
1 : DEFINE CHANNEL(INVENTORY.NEWYORK) CHLTYPE(CLUSSDR) TRPTYPE(LU62)
CONNAME(NEWYORK.LUNAME) CLUSTER(INVENTORY)
MODENAME('#INTER') TPNAME('MQSERIES')
DESCR('LU62 Cluster-sender channel from LONDON to repository at NEWYORK')
AMQ8014: WebSphere MQ channel created.
07/09/98 13:00:18  Channel program started.
```

8. Define the CLUSSDR channel on the `NEWYORK` queue manager.

   ```
   DEFINE CHANNEL(INVENTORY.LONDON) CHLTYPE(CLUSSDR) TRPTYPE(LU62)
   CONNAME(LONDON.LUNAME) CLUSTER(INVENTORY)
   DESCR('LU62 Cluster-sender channel from NEWYORK to repository at LONDON')
   ```
9. Define the cluster queue `INVENTQ`

   Define the `INVENTQ` queue on the `NEWYORK` queue manager, specifying the `CLUSTER` keyword.

   ```
   DEFINE QLOCAL(INVENTQ) CLUSTER(INVENTORY)
   ```

```
1 : DEFINE QLOCAL(INVENTQ) CLUSTER(INVENTORY)
AMQ8006: WebSphere MQ queue created.
```

The `CLUSTER` keyword causes the queue to be advertised to the cluster. As soon as the queue is defined it becomes available to the other queue managers in the cluster. They can send messages to it without having to make a remote-queue definition for it.

All the definitions are complete. On all platforms, start a listener program on each queue manager. The listener program waits for incoming network requests and starts the cluster-receiver channel when it is needed.

Verifying the cluster:
About this task

You can verify the cluster in one or more of these ways:
1. Running administrative commands to display cluster and channel attributes.
2. Run the sample programs to send and receive messages on a cluster queue.
3. Write your own programs to send a request message to a cluster queue and reply with a response messages to an non-clustered reply queue.

Procedure
1. Issue `DISPLAY` **`runmqsc`** commands to verify the cluster. The responses you see ought to be like the responses in the steps that follow.
   a. 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)
```

b. 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.QUEUE) XMITQ(SYSTEM.CLUSTER.TRANSMIT.INVENTORY.LONDON)
CONNAME(192.0.2.1)      CURRENT
CHLTYPE(CLUSSDR)       STATUS(RUNNING)
RQMNAME(LONDON)
```

2. Send messages between the two queue managers, using **amqsput**.
   a. On LONDON run the command **amqsput INVENTQ LONDON**.

      Type some messages, followed by a blank line.
   b. On NEWYORK run the command **amqsget INVENTQ NEWYORK**.

      You now see the messages you entered on LONDON. After 15 seconds the program ends.
3. 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.
   a. On LONDON put a messages to the cluster queue.
      1) Define a local queue called LONDON_reply
      2) Set the MQOPEN options to MQOO_OUTPUT
      3) Issue the MQOPEN call to open the queue INVENTQ
      4) Set the *ReplyToQ* name in the message descriptor to LONDON_reply
      5) Issue the MQPUT call to put the message
      6) Commit the message
   b. On NEWYORK receive the message on the cluster queue and put a reply to the reply queue.
      1) Set the MQOPEN options to MQOO_BROWSE
      2) Issue the MQOPEN call to open the queue INVENTQ
      3) Issue the MQGET call to get the message from INVENTQ
      4) Retrieve the *ReplyToQ* name from the message descriptor
      5) Put the *ReplyToQ* name in the ObjectName field of the object descriptor
      6) Set the MQOPEN options to MQOO_OUTPUT
      7) Issue the MQOPEN call to open LONDON_reply at queue manager LONDON
      8) Issue the MQPUT call to put the message to LONDON_reply
   c. On LONDON receive the reply.
      1) Set the MQOPEN options to MQOO_BROWSE
      2) Issue the MQOPEN call to open the queue LONDON_reply

3) 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 1074. It contains two queue managers, LONDON and NEWYORK, which both hold full repositories.
- The queue manager PARIS is owned by the primary installation. If it is not, you must run the **setmqenv** command to set up the command environment for the installation that PARIS belongs to.
- TCP connectivity exists between all three systems, and the queue manager is configured with a TCP listener that starts under the control of the queue manager.

About this task

1. A new branch of the chain store is being set up in Paris and you want to add a queue manager called PARIS to the cluster.
2. Queue manager PARIS sends inventory updates to the application running on the system in New York by putting messages on the INVENTQ queue.

Follow these steps to add a queue manager to a cluster.

Procedure

1. Decide which full repository PARIS refers to first.

   Every queue manager in a cluster must refer to one or other of the full repositories. It gathers information about the cluster from a full repository and so builds up its own partial repository. Choose either of the repositories as the full repository. As soon as a new queue manager is added to the cluster it immediately learns about the other repository as well. Information about changes to a queue manager is sent directly to two repositories. In this example, you link PARIS to the queue manager LONDON, purely for geographical reasons.

   Note: Perform the remaining steps in any order, after queue manager PARIS is started.
2. Define a CLUSRCVR channel on queue manager PARIS.

   Every queue manager in a cluster must define a cluster-receiver channel on which it can receive messages. On PARIS, define:

   ```
   DEFINE CHANNEL(INVENTORY.PARIS) CHLTYPE(CLUSRCVR) TRPTYPE(TCP)
   CONNAME(PARIS.CHSTORE.COM) CLUSTER(INVENTORY)
   DESCR('Cluster-receiver channel for queue manager PARIS')
   ```

   The cluster-receiver channel advertises the availability of the queue manager to receive messages from other queue managers in the cluster INVENTORY. Do not make definitions on other queue managers for a sending end to the cluster-receiver channel INVENTORY.PARIS. Other definitions are made automatically when needed. See Cluster channels.
3. z/OS Start the channel initiator on IBM MQ for z/OS.
4. Define a CLUSSDR channel on queue manager PARIS.

   When you add to a cluster a queue manager that is not a full repository, you define just one cluster-sender channel to make an initial connection to a full repository. See Cluster-sender channel: CLUSSDR . On PARIS, make the following definition for a CLUSSDR channel called INVENTORY.LONDON to the queue manager with the network address LONDON.CHSTORE.COM.

```
DEFINE CHANNEL(INVENTORY.LONDON) CHLTYPE(CLUSSDR) TRPTYPE(TCP)
CONNAME(LONDON.CHSTORE.COM) CLUSTER(INVENTORY)
DESCR('Cluster-sender channel from PARIS to repository at LONDON')
```

5. Optional: If you are adding to a cluster a queue manager that has previously been removed from the same cluster, check that it is now showing as a cluster member. If not, complete the following extra steps:
   a. Issue the **REFRESH CLUSTER** command on the queue manager you are adding. This step stops the cluster channels, and gives your local cluster cache a fresh set of sequence numbers that are assured to be up-to-date within the rest of the cluster.

      ```
      REFRESH CLUSTER(INVENTORY) REPOS(YES)
      ```

      Note: For large clusters, using the **REFRESH CLUSTER** command can be disruptive to the cluster while it is in progress, and again at 27 day intervals thereafter when the cluster objects automatically send status updates to all interested queue managers. See Refreshing in a large cluster can affect performance and availability of the cluster.
   b. Restart the CLUSSDR channel (for example, using the START CHANNEL command).
   c. Restart the CLUSRCVR channel.

Results

The following figure shows the cluster set up by this task.

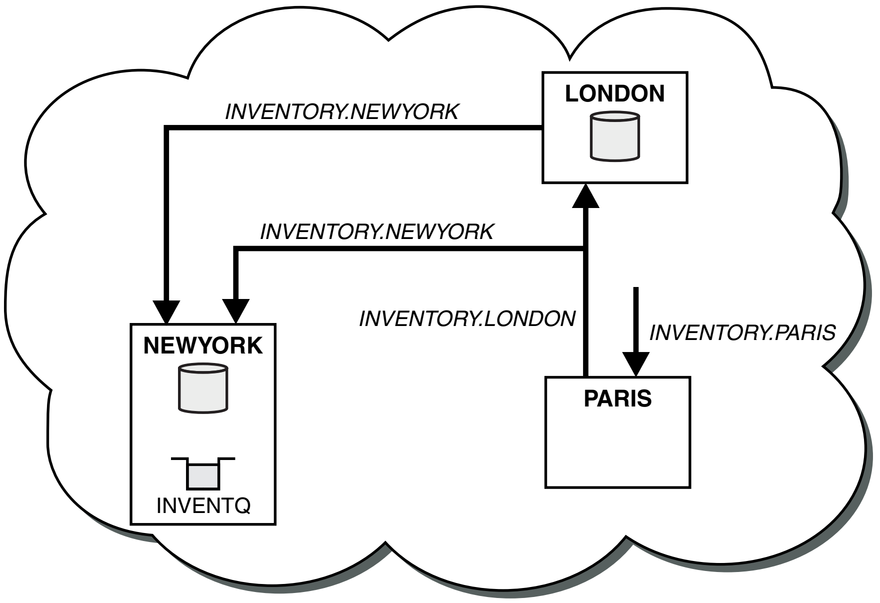


Figure 118. The `INVENTORY` cluster with three queue managers

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 1074.

About this task

This task is an alternative to "Adding a queue manager to a cluster" on page 1085, 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`*.
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.

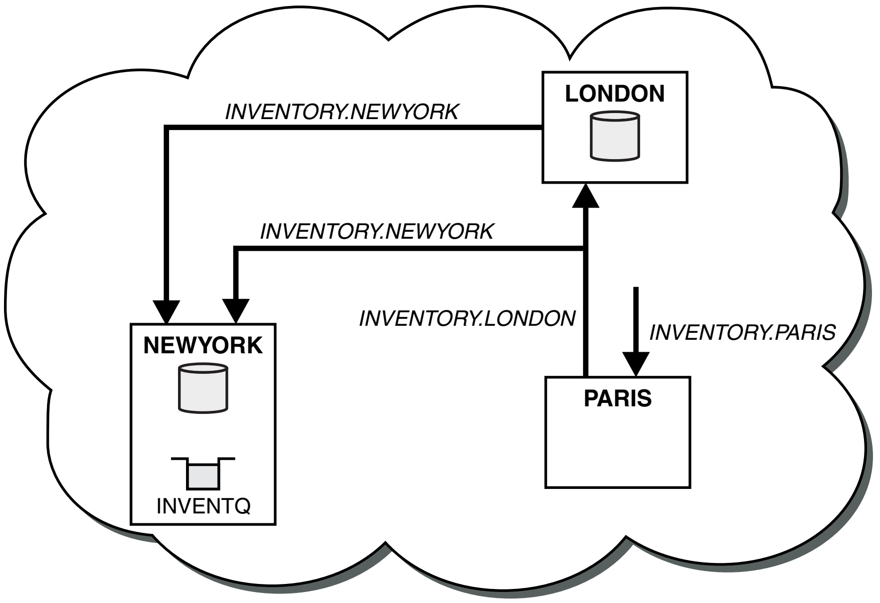


Figure 119. The INVENTORY cluster with three queue managers

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 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 1074. 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 1085:

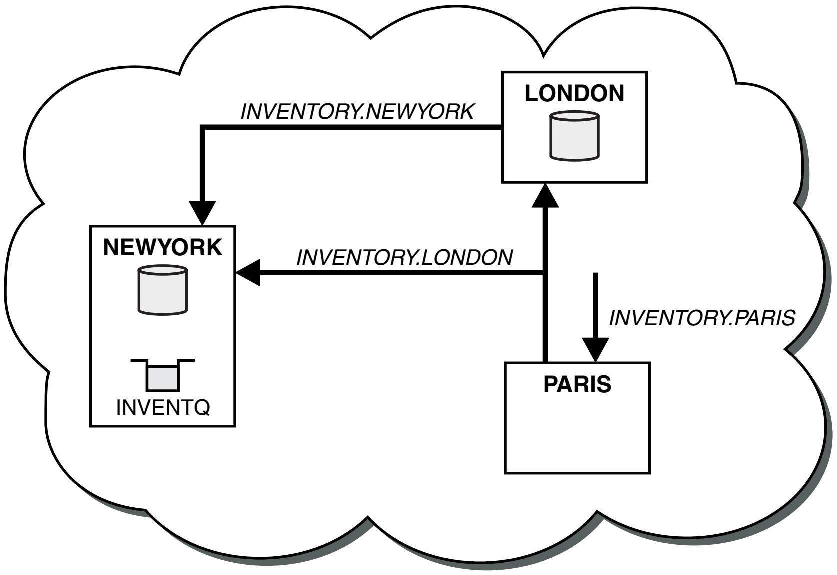


Figure 120. The INVENTORY cluster with three queue managers

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 information:
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 1085. 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 1096.

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 121 shows the `INVENTORY` cluster set up by this task.

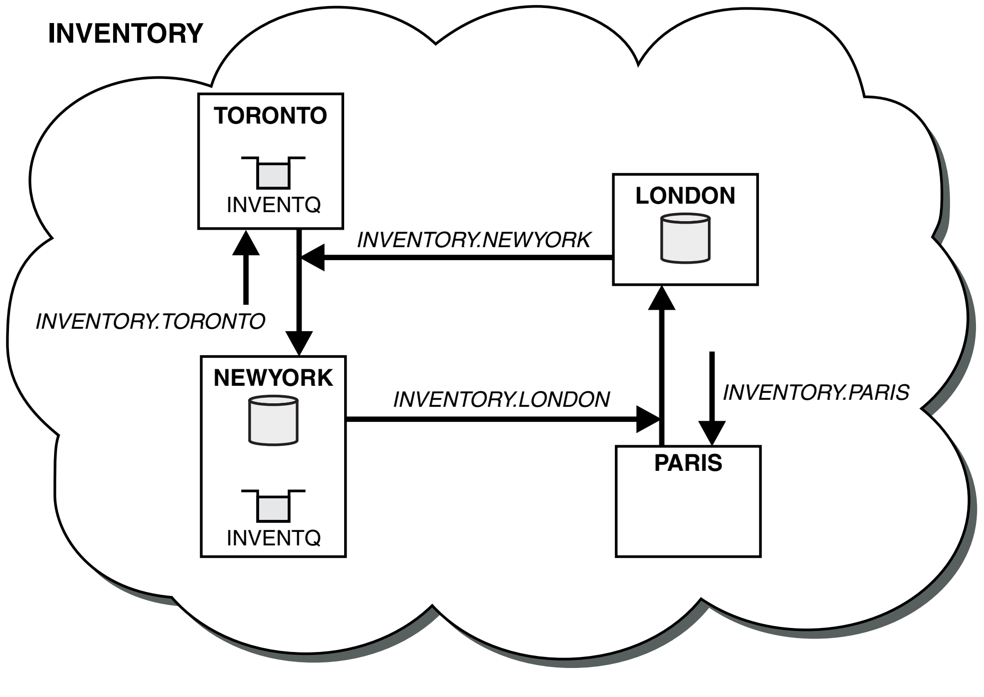


Figure 121. The `INVENTORY` cluster with four queue managers

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

z/OS

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 1074. 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 122 shows the cluster set up by this task.

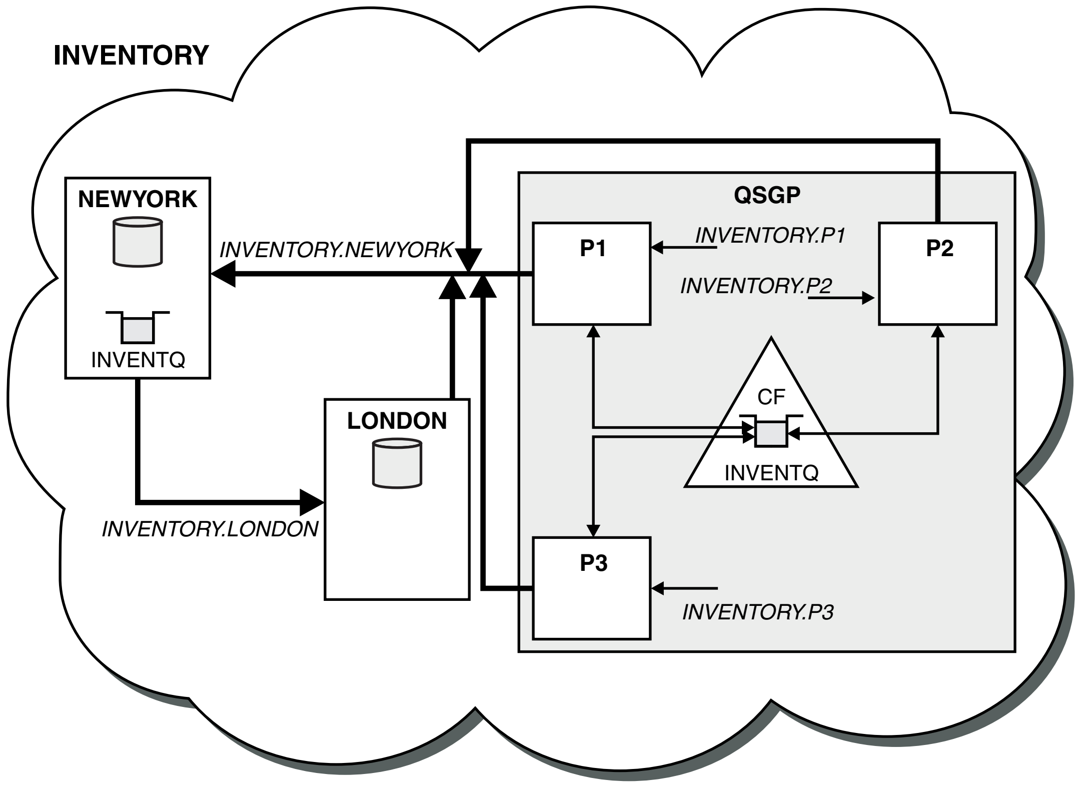


Figure 122. Cluster and queue-sharing group

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 1085.
- 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 123 on page 1098 shows the cluster set up by this task.

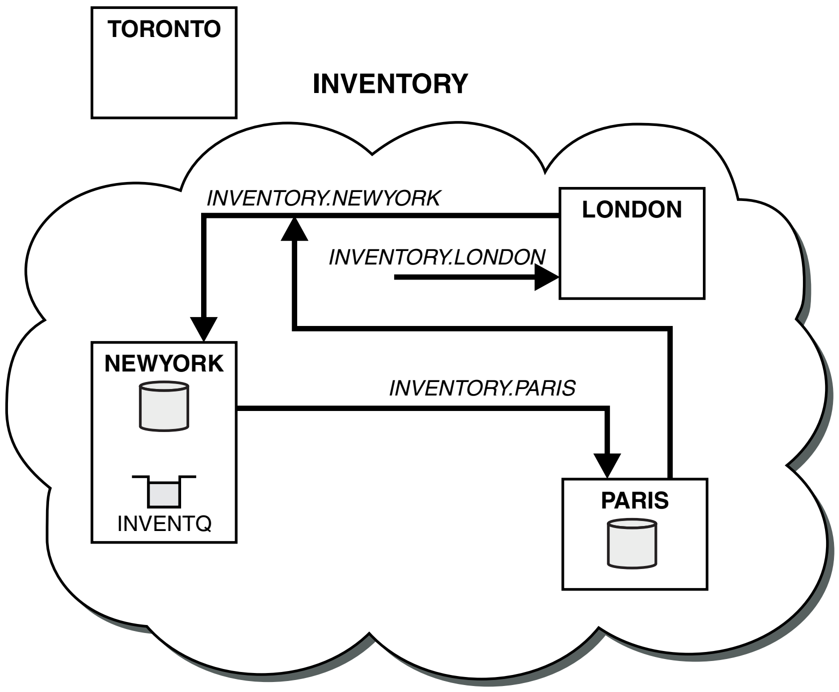


Figure 123. The `INVENTORY` cluster with the full repository moved to `PARIS`

Establishing communication in a cluster

A channel initiator is needed to start a communication channel when there is a message to deliver. A channel listener waits to start the other end of a channel to receive the message.

Before you begin

To establish communication between queue managers in a cluster, configure a link using one of the supported communication protocols. The supported protocols are TCP or LU 6.2 on any platform, and NetBIOS or SPX on Windows systems. As part of this configuration, you also need channel initiators and channel listeners just as you do with distributed queuing.

About this task

All cluster queue managers need a channel initiator to monitor the system-defined initiation queue `SYSTEM.CHANNEL.INITQ`. `SYSTEM.CHANNEL.INITQ` is the initiation queue for all transmission queues including the cluster transmission queue.

Each queue manager must have a channel listener. A channel listener program waits for incoming network requests and starts the appropriate receiver-channel when it is needed. The implementation of channel listeners is platform-specific, however there are some common features. On all IBM MQ platforms, the listener can be started using the `START LISTENER` command. On IBM MQ for IBM i, Windows, UNIX and Linux systems, you can start the listener automatically at the same time as the queue manager. To start the listener automatically, set the `CONTROL` attribute of the `LISTENER` object to `QMGR` or `STARTONLY`.

Procedure

1. Start the channel initiator.
   - z/OS

IBM MQ for z/OS
: There is one channel initiator for each queue manager and it runs as a separate address space. You start it using the **MQSC** `START CHINIT` command, which you issue as part of your queue manager startup.

- ULW

  IBM MQ for UNIX, 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

  IBM MQ for IBM i
  When you start a queue manager, if the queue manager attribute `SCHINIT` is set to `QMGR`, a channel initiator is automatically started. Otherwise it can be started using the **runmqsc** `START CHINIT` command or the **runmqchi** control command.

2. Start the channel listener.

- z/OS

  IBM MQ for z/OS
  Use the channel listener program provided by IBM MQ. To start an IBM MQ channel listener, use the **MQSC** command `START LISTENER`, which you issue as part of your channel initiator startup. For example:

  ```
  START LISTENER PORT(1414) TRPTYPE(TCP)
  ```

  or:

  ```
  START LISTENER LUNAME(LONDON.LUNAME) TRPTYPE(LU62)
  ```

  Members of a queue-sharing group can use a shared listener instead of a listener for each queue manager. Do not use shared listeners with clusters. Specifically, do not make the `CONNAME` of the `CLUSRCVR` channel the address of the shared listener of the queue-sharing group. If you do, queue managers might receive messages for queues for which they do not have a definition.

- IBM i

  IBM MQ for IBM i
  Use the channel listener program provided by IBM MQ. To start an IBM MQ channel listener use the **CL** command `STRMQMLSR`. For example:

  ```
  STRMQMLSR MQMNAME(QM1) PORT(1414)
  ```

- Windows

  IBM MQ for Windows
  Use either the channel listener program provided by IBM MQ, or the facilities provided by the operating system.

  To start the IBM MQ channel listener use the `RUNMQLSR` command. For example:

  ```
  RUNMQLSR -t tcp -p 1414 -m QM1
  ```

- UNIX Linux

  IBM MQ on UNIX 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 **runmqlsr** command. For example:

  ```
  runmqlsr -t tcp -p 1414 -m QM1
  ```

To use **inetd** to start channels, configure two files:

a. Edit the file /etc/services. You must be logged in as a superuser or root. If the following line is not in the file, add it as shown:

```
MQSeries    1414/tcp   # WebSphere MQ channel listener
```

where 1414 is the port number required by IBM MQ. You can change the port number, but it must match the port number specified at the sending end.

b. Edit the file /etc/inetd.conf. If you do not have the following line in that file, add it as shown:

```
MQSeries stream tcp nowait mqm MQ_INSTALLATION_PATH/bin/amqcrsta amqcrsta
-m queue.manager.name
```

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

The updates become active after **inetd** has reread the configuration files. Issue the following commands from the root user ID:

On AIX:

```
refresh -s inetd
```

On HP-UX:

```
inetd -c
```

On Solaris or Linux:

a. Find the process ID of the **inetd** with the command:

```
ps -ef | grep inetd
```

b. Run the appropriate command, as follows:
   - For Solaris 9 and Linux:

     ```
     kill -1 inetd processid
     ```

   - For Solaris 10, or later versions:

     ```
     inetconv
     ```

Converting an existing network into a cluster

Convert an existed distributed queuing network to a cluster and add an additional queue manager to increase capacity.

Before you begin

In “Setting up a new cluster” on page 1074 through “Moving a full repository to another queue manager” on page 1096 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 124 on page 1101.

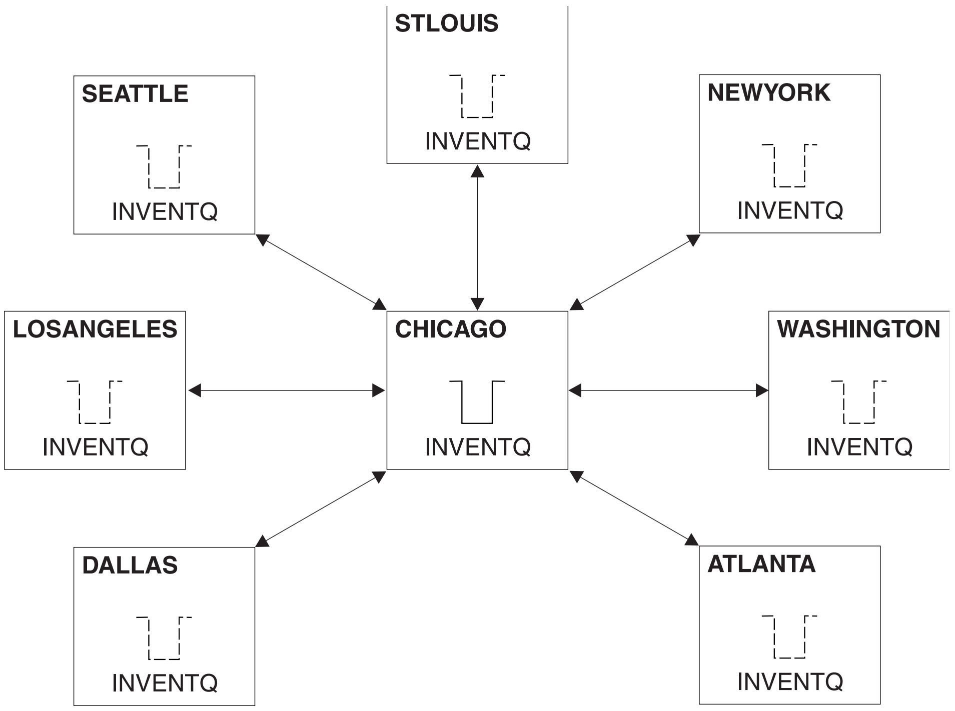


Figure 124. A hub and spoke network

- 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 1162
2. Alter the two central queue managers to make them full repository queue managers.

The two queue managers CHICAGO and CHICAGO2 are at the hub of this network. You have decided to concentrate all activity associated with the chain store cluster on to those two queue managers. As well as the inventory application and the definitions for the INVENTQ queue, you want these queue managers to host the two full repositories for the cluster. At each of the two queue managers, issue the following command:

```
ALTER QMGR REPOS(CHNSTORE)
```

3. Define a CLUSRCVR channel on each queue manager.

   At each queue manager in the cluster, define a cluster-receiver channel and a cluster-sender channel. It does not matter which channel you define first.

   Make a CLUSRCVR definition to advertise each queue manager, its network address, and other information, to the cluster. For example, on queue manager ATLANTA:

   ```
   DEFINE CHANNEL(CHNSTORE.ATLANTA) CHLTYPE(CLUSRCVR) TRPTYPE(TCP)
   CONNAME(ATLANTA.CHSTORE.COM) CLUSTER(CHNSTORE)
   DESCR('Cluster-receiver channel')
   ```
4. Define a CLUSSDR channel on each queue manager

   Make a CLUSSDR definition at each queue manager to link that queue manager to one or other of the full repository queue managers. For example, you might link ATLANTA to CHICAGO2:

   ```
   DEFINE CHANNEL(CHNSTORE.CHICAGO2) CHLTYPE(CLUSSDR) TRPTYPE(TCP)
   CONNAME(CHICAGO2.CHSTORE.COM) CLUSTER(CHNSTORE)
   DESCR('Cluster-sender channel to repository queue manager')
   ```
5. Install the inventory application on CHICAGO2.

   You already have the inventory application on queue manager CHICAGO. Now you need to make a copy of this application on queue manager CHICAGO2.
6. Define the INVENTQ queue on the central queue managers.

   On CHICAGO, modify the local queue definition for the queue INVENTQ to make the queue available to the cluster. Issue the command:

   ```
   ALTER QLOCAL(INVENTQ) CLUSTER(CHNSTORE)
   ```

   On CHICAGO2, make a definition for the same queue:

   ```
   DEFINE QLOCAL(INVENTQ) CLUSTER(CHNSTORE)
   ```

   On z/OS, you can use the MAKEDEF option of the COMMAND function of **CSQUTIL** to make an exact copy on CHICAGO2 of the INVENTQ on CHICAGO.

   When you make these definitions, a message is sent to the full repositories at CHICAGO and CHICAGO2 and the information in them is updated. The queue manager finds out from the full repositories when it puts a message to the INVENTQ, that there is a choice of destinations for the messages.
7. Check that the cluster changes have been propagated.

   Check that the definitions you created in the previous step have been propagated though the cluster. Issue the following command on a full repository queue manager:

   ```
   DIS QCLUSTER(INVENTQ)
   ```

Adding a new, interconnected cluster:

Add a new cluster that shares some queue managers with an existing cluster.

Before you begin

Note:

1. For changes to a cluster to be propagated throughout the cluster, at least one full repository must always be available. Ensure that your repositories are available before starting this task.
2. Before starting this task, check for queue-name clashes and understand the consequences. You might need to rename a queue, or set up queue aliases before you can proceed.

Scenario:

- An IBM MQ cluster has been set up as described in "Converting an existing network into a cluster" on page 1100.
- A new cluster called `MAILORDER` is to be implemented. This cluster comprises four of the queue managers that are in the `CHNSTORE` cluster; `CHICAGO`, `CHICAGO2`, `SEATTLE`, and `ATLANTA`, and two additional queue managers; `HARTFORD` and `OMAHA`. The `MAILORDER` application runs on the system at Omaha, connected to queue manager `OMAHA`. It is driven by the other queue managers in the cluster putting messages on the `MORDERQ` queue.
- The full repositories for the `MAILORDER` cluster are maintained on the two queue managers `CHICAGO` and `CHICAGO2`.
- The network protocol is TCP.

About this task

Follow these steps to add a new, interconnected cluster.

Procedure

1. Create a namelist of the cluster names.

   The full repository queue managers at `CHICAGO` and `CHICAGO2` are now going to hold the full repositories for both of the clusters `CHNSTORE` and `MAILORDER`. First, create a namelist containing the names of the clusters. Define the namelist on `CHICAGO` and `CHICAGO2`, as follows:

   ```
   DEFINE NAMELIST(CHAINMAIL)
   DESCR('List of cluster names')
   NAMES(CHNSTORE, MAILORDER)
   ```
2. Alter the two queue manager definitions.

   Now alter the two queue manager definitions at `CHICAGO` and `CHICAGO2`. Currently these definitions show that the queue managers hold full repositories for the cluster `CHNSTORE`. Change that definition to show that the queue managers hold full repositories for all clusters listed in the `CHAINMAIL` namelist. Alter the `CHICAGO` and `CHICAGO2` queue manager definitions:

   ```
   ALTER QMGR REPOS(' ') REPOSNL(CHAINMAIL)
   ```
3. Alter the `CLUSRCVR` channels on `CHICAGO` and `CHICAGO2`.

   The `CLUSRCVR` channel definitions at `CHICAGO` and `CHICAGO2` show that the channels are available in the cluster `CHNSTORE`. You need to change the cluster-receiver definition to show that the channels are available to all clusters listed in the `CHAINMAIL` namelist. Change the cluster-receiver definition at `CHICAGO`:

   ```
   ALTER CHANNEL(CHNSTORE.CHICAGO) CHLTYPE(CLUSRCVR)
   CLUSTER(' ') CLUSNL(CHAINMAIL)
   ```

   At `CHICAGO2`, enter the command:

   ```
   ALTER CHANNEL(CHNSTORE.CHICAGO2) CHLTYPE(CLUSRCVR)
   CLUSTER(' ') CLUSNL(CHAINMAIL)
   ```

4. Alter the CLUSSDR channels on CHICAGO and CHICAGO2.

   Change the two CLUSSDR channel definitions to add the namelist. At CHICAGO, enter the command:

   ```
   ALTER CHANNEL(CHNSTORE.CHICAGO2) CHLTYPE(CLUSSDR)
   CLUSTER(' ') CLUSNL(CHAINMAIL)
   ```

   At CHICAGO2, enter the command:

   ```
   ALTER CHANNEL(CHNSTORE.CHICAGO) CHLTYPE(CLUSSDR)
   CLUSTER(' ') CLUSNL(CHAINMAIL)
   ```

5. Create a namelist on SEATTLE and ATLANTA.

   Because SEATTLE and ATLANTA are going to be members of more than one cluster, you must create a namelist containing the names of the clusters. Define the namelist on SEATTLE and ATLANTA, as follows:

   ```
   DEFINE NAMELIST(CHAINMAIL)
   DESCR('List of cluster names')
   NAMES(CHNSTORE, MAILORDER)
   ```

6. Alter the CLUSRCVR channels on SEATTLE and ATLANTA.

   The CLUSRCVR channel definitions at SEATTLE and ATLANTA show that the channels are available in the cluster CHNSTORE. Change the cluster-receive channel definitions to show that the channels are available to all clusters listed in the CHAINMAIL namelist. At SEATTLE, enter the command:

   ```
   ALTER CHANNEL(CHNSTORE.SEATTLE) CHLTYPE(CLUSRCVR)
   CLUSTER(' ') CLUSNL(CHAINMAIL)
   ```

   At ATLANTA, enter the command:

   ```
   ALTER CHANNEL(CHNSTORE.ATLANTA) CHLTYPE(CLUSRCVR)
   CLUSTER(' ') CLUSNL(CHAINMAIL)
   ```

7. Alter the CLUSSDR channels on SEATTLE and ATLANTA.

   Change the two CLUSSDR channel definitions to add the namelist. At SEATTLE, enter the command:

   ```
   ALTER CHANNEL(CHNSTORE.CHICAGO) CHLTYPE(CLUSSDR)
   CLUSTER(' ') CLUSNL(CHAINMAIL)
   ```

   At ATLANTA, enter the command:

   ```
   ALTER CHANNEL(CHNSTORE.CHICAGO2) CHLTYPE(CLUSSDR)
   CLUSTER(' ') CLUSNL(CHAINMAIL)
   ```

8. Define CLUSRCVR and CLUSSDR channels on HARTFORD and OMAHA.

   On the two new queue managers HARTFORD and OMAHA, define cluster-receiver and cluster-sender channels. It does not matter in which sequence you make the definitions. At HARTFORD, enter:

   ```
   DEFINE CHANNEL(MAILORDER.HARTFORD) CHLTYPE(CLUSRCVR) TRPTYPE(TCP)
   CONNAME(HARTFORD.CHSTORE.COM) CLUSTER(MAILORDER)
   DESCR('Cluster-receiver channel for HARTFORD')

   DEFINE CHANNEL(MAILORDER.CHICAGO) CHLTYPE(CLUSSDR) TRPTYPE(TCP)
   CONNAME(CHICAGO.CHSTORE.COM) CLUSTER(MAILORDER)
   DESCR('Cluster-sender channel from HARTFORD to repository at CHICAGO')
   ```

   At OMAHA, enter:

   ```
   DEFINE CHANNEL(MAILORDER.OMAHA) CHLTYPE(CLUSRCVR) TRPTYPE(TCP)
   CONNAME(OMAHA.CHSTORE.COM) CLUSTER(MAILORDER)
   DESCR('Cluster-receiver channel for OMAHA')

   DEFINE CHANNEL(MAILORDER.CHICAGO) CHLTYPE(CLUSSDR) TRPTYPE(TCP)
   CONNAME(CHICAGO.CHSTORE.COM) CLUSTER(MAILORDER)
   DESCR('Cluster-sender channel from OMAHA to repository at CHICAGO')
   ```

9. Define the MORDERQ queue on OMAHA.

   The final step to complete this task is to define the queue MORDERQ on the queue manager OMAHA. At OMAHA, enter:

   ```
   DEFINE QLOCAL(MORDERQ) CLUSTER(MAILORDER)
   ```

10. Check that the cluster changes have been propagated.

Check that the definitions you created with the previous steps have been propagated though the cluster. Issue the following commands on a full repository queue manager:

```
DIS QCLUSTER (MORDERQ)
DIS CLUSQMGR
```

11.

Results

The cluster set up by this task is shown in Figure 125 on page 1106.

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.

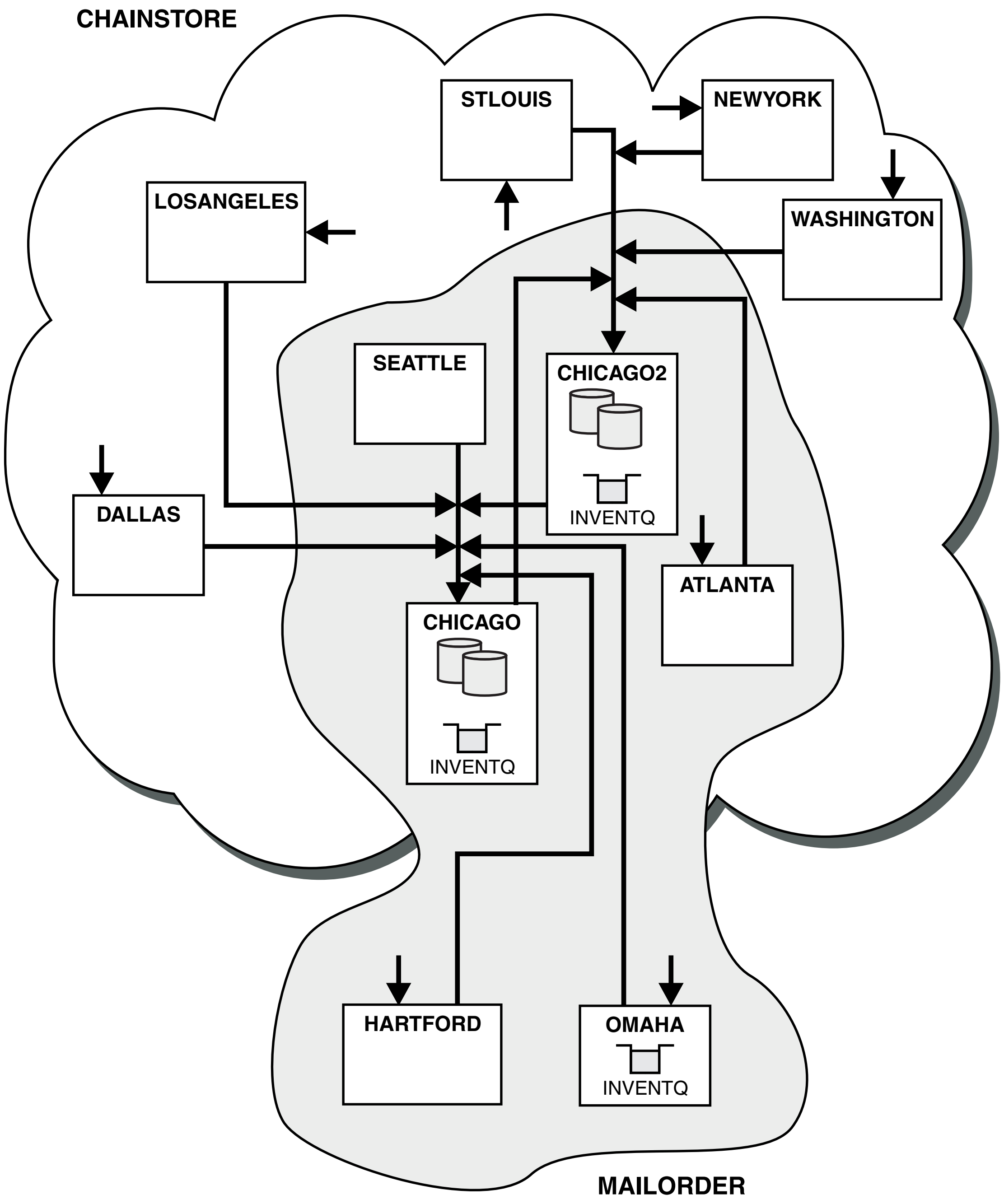


Figure 125. Interconnected clusters

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.

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 1100.
- 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 1132.
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.

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 126 on page 1109. The example is described in Clustering: Application isolation using multiple cluster transmission queues.

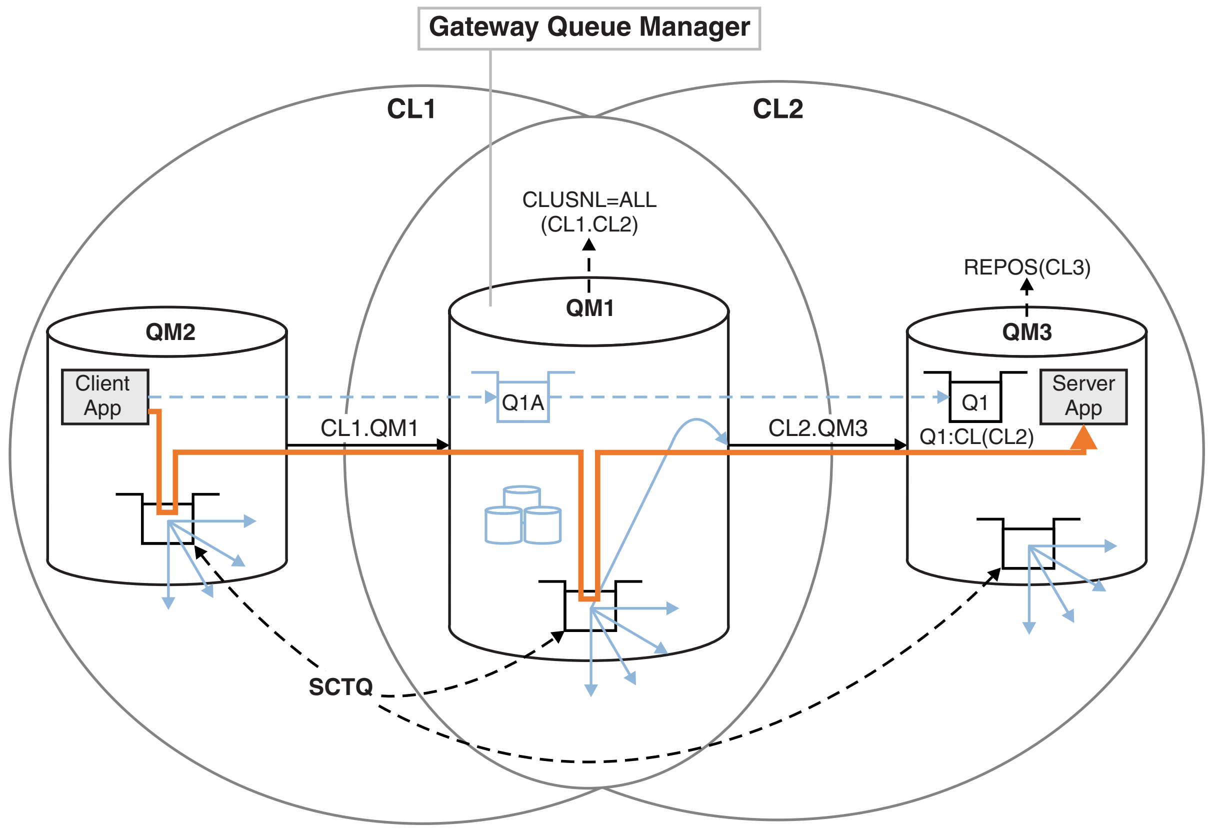


Figure 126. Client-server application deployed to hub and spoke architecture using IBM MQ clusters

To make the number of steps to construct the example as few as possible, the configuration is kept simple, rather than realistic. The example might represent the integration of two clusters created by two separate organizations. For a more realistic scenario, see Clustering: Planning how to configure cluster transmission queues.

Follow the steps to construct the clusters. The clusters are used in the following examples of isolating the message traffic from the client application to the server application.

The instructions add a couple of extra queue managers so that each cluster has two repositories. The gateway queue manager is not used as a repository for performance reasons.

Procedure

1. Create and start the queue managers QM1, QM2, QM3, QM4, QM5.

   ```
   crtmqm -sax -u SYSTEM.DEAD.LETTER.QUEUE QM n
   strmqm QmgrName
   ```

   Note: QM4 and QM5 are the backup full repositories for the clusters.
2. Define and start listeners for each of the queue managers.

   ```
   *... On QM n
   DEFINE LISTENER(TCP141 n) TRPTYPE(TCP) IPADDR(hostname) PORT(141 n) CONTROL(QMGR) REPLACE
   START LISTENER(TCP141 n)
   ```
3. Create a cluster name list for all of the clusters.

   ```
   *... On QM1
   DEFINE NAMELIST(ALL) NAMES(CL1, CL2) REPLACE
   ```
4. Make QM2 and QM4 full repositories for CL1, QM3 and QM5 full repositories for CL2.

a. For CL1:

```
*... On QM2 and QM4
ALTER QMGR REPOS(CL1) DEFCLXQ(SCTQ)
```

b. For CL2:

```
*... On QM3 and QM5
ALTER QMGR REPOS(CL2) DEFCLXQ(SCTQ)
```

5. Add the cluster-sender and cluster-receiver channels for each queue manager and cluster.

   Run the following commands on QM2, QM3, QM4 and QM5, where *c*, *n*, and *m* take the values shown in Table 122 for each queue manager:

Table 122. Parameter values for creating clusters 1 and 2

Queue manager	Cluster *c*	Other repository *n*	This repository *m*
QM2	1	4	2
QM4	1	2	4
QM3	2	5	3
QM5	2	3	5

```
*... On QM m
DEFINE CHANNEL(CL c.QM n) CHLTYPE(CLUSSDR) CONNAME('localhost(141 n)') CLUSTER(CL c) REPLACE
DEFINE CHANNEL(CL c.QM m) CHLTYPE(CLUSRCVR) CONNAME('localhost(141 m)') CLUSTER(CL c) REPLACE
```

6. Add the gateway queue manager, QM1, to each of the clusters.

```
*... On QM1
DEFINE CHANNEL(CL1.QM2) CHLTYPE(CLUSSDR) CONNAME('localhost(1412)') CLUSTER(CL1) REPLACE
DEFINE CHANNEL(CL1.QM1) CHLTYPE(CLUSRCVR) CONNAME('localhost(1411)') CLUSTER(CL1) REPLACE
DEFINE CHANNEL(CL2.QM3) CHLTYPE(CLUSSDR) CONNAME('localhost(1413)') CLUSTER(CL2) REPLACE
DEFINE CHANNEL(CL2.QM1) CHLTYPE(CLUSRCVR) CONNAME('localhost(1411)') CLUSTER(CL2) REPLACE
```

7. Add the local queue Q1 to queue manager QM3 in cluster CL2.

```
*... On QM3
DEFINE QLOCAL(Q1) CLUSTER(CL2) REPLACE
```

8. Add the clustered queue manager alias Q1A to the gateway queue manager.

```
*... On QM1
DEFINE QALIAS(Q1A) CLUSNL(ALL) TARGET(Q1) TARGTYPE(QUEUE) DEFBIND(NOTFIXED) REPLACE
```

   Note: Applications using the queue manager alias on any other queue manager but QM1, must specify DEFBIND(NOTFIXED) when they open the alias queue. **DEFBIND** specifies whether the routing information in the message header is fixed when the queue is opened by the application. If it is set to the default value, OPEN, messages are routed to Q1@QM1. Q1@QM1 does not exist, so messages from other queue managers end up on a dead letter queue. By setting the queue attribute to DEFBIND(NOTFIXED), applications such as **amqsput**, which default to the queue setting of **DEFBIND**, behave in the correct way.

9. Add the cluster queue manager alias definitions for all the clustered queue managers to the gateway queue manager, QM1.

```
*... On QM1
DEFINE QREMOTE(QM2) RNAME(' ') RQMNAME(QM2) CLUSNL(ALL) REPLACE
DEFINE QREMOTE(QM3) RNAME(' ') RQMNAME(QM3) CLUSNL(ALL) REPLACE
```

   Tip: The queue manager alias definitions on the gateway queue manager transfer messages that refer to a queue manager in another cluster; see Clustered queue manager aliases.

What to do next

1. Test the queue alias definition by sending a message from QM2 to Q1 on QM3 using the queue alias definition Q1A.

   a. Run the sample program **amqsput** on QM2 to put a message.

```
C:\IBM\MQ>amqsput Q1A QM2
Sample AMQSPUT0 start
target queue is Q1A
Sample request message from QM2 to Q1 using Q1A

Sample AMQSPUT0 end
```

b. Run the sample program **amqsget** to get the message from Q1 on QM3

```
C:\IBM\MQ>amqsget Q1 QM3
Sample AMQSGET0 start
message <Sample request message from QM2 to Q1 using Q1A>
no more messages
Sample AMQSGET0 end
```

2. Test the queue manager alias definitions by sending a request message and receiving a reply message on a temporary-dynamic reply queue.

   The diagram shows the path taken by the reply message back to a temporary dynamic queue, which is called RQ. The server application, connected to QM3, opens the reply queue using the queue manager name QM2. The queue manager name QM2 is defined as a clustered queue manager alias on QM1. QM3 routes the reply message to QM1. QM1 routes the message to QM2.

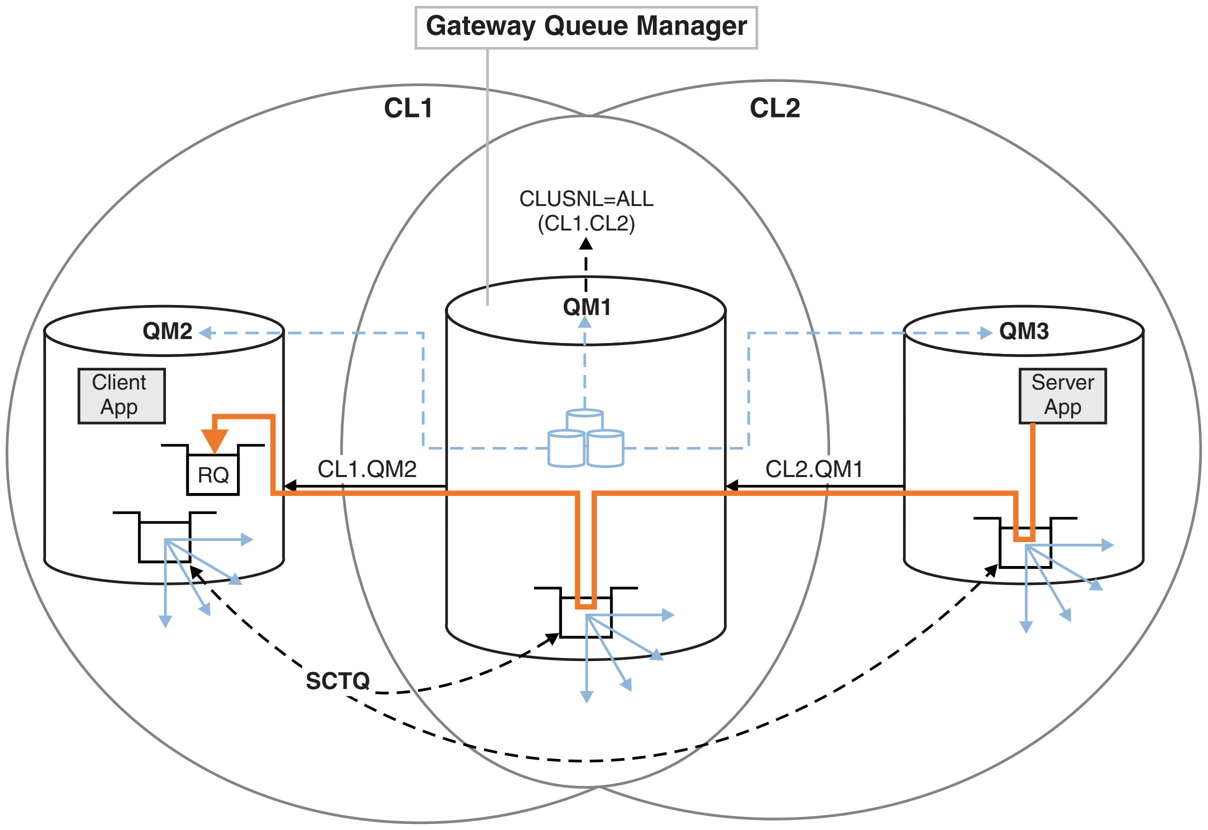


Figure 127. Using a queue manager alias to return the reply message to a different cluster

The way the routing works is as follows. Every queue manager in each cluster has a queue manager alias definition on QM1. The aliases are clustered in all the clusters. The grey dashed arrows from each of the aliases to a queue manager show that each queue manager alias is resolved to a real queue manager in at least one of the clusters. In this case, the QM2 alias is clustered in both cluster CL1 and CL2, and is resolved to the real queue manager QM2 in CL1. The server application creates the reply message using the reply to queue name RQ, and reply to queue manager name QM2. The message is routed to QM1 because the queue manager alias definition QM2 is defined on QM1 in cluster CL2 and queue manager QM2 is not in cluster CL2. As the message cannot be sent to the target queue manager, it is sent to the queue manager that has the alias definition.

QM1 places the message on the cluster transmission queue on QM1 for transferal to QM2. QM1 routes the message to QM2 because the queue manager alias definition on QM1 for QM2 defines QM2 as the real target queue manager. The definition is not circular, because alias definitions can refer only to real definitions; the alias cannot point to itself. The real definition is resolved by QM1, because both QM1 and QM2 are in the same cluster, CL1. QM1 finds out the connection information for QM2 from the repository for CL1, and routes the message to QM2. For the message to be rerouted by QM1, the server application must have opened the reply queue with the option DEFBIND set to MQBND_BIND_NOT_FIXED. If the server application had opened the reply queue with the option MQBND_BIND_ON_OPEN, the message is not rerouted and ends up on a dead letter queue.

a. Create a clustered request queue with a trigger on QM3.

   ```
   *... On QM3
   DEFINE QLOCAL(QR) CLUSTER(CL2) TRIGGER INITQ(SYSTEM.DEFAULT.INITIATION.QUEUE) PROCESS(ECHO) REPLACE
   ```

b. Create a clustered queue alias definition of QR on the gateway queue manager, QM1.

   ```
   *... On QM1
   DEFINE QALIAS(QRA) CLUSNL(ALL) TARGET(QR) TARGTYPE(QUEUE) DEFBIND(NOTFIXED) REPLACE
   ```

c. Create a process definition to start the sample echo program **amqsech** on QM3.

   ```
   *... On QM3
   DEFINE PROCESS(ECHO) APPLICID(AMQSECH) REPLACE
   ```

d. Create a model queue on QM2 for the sample program **amqsreq** to create the temporary-dynamic reply queue.

   ```
   *... On QM2
   DEFINE QMODEL(SYSTEM.SAMPLE.REPLY) REPLACE
   ```

e. Test the queue manager alias definition by sending a request from QM2 to QR on QM3 using the queue alias definition QRA.

   1) 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, 2011. 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
      ```

   2) 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 tasks:

"Adding a queue manager to a cluster: separate transmission queues" on page 1087
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 information:

Access control and multiple cluster transmission queues

Clustering: Application isolation using multiple cluster transmission queues

Clustering: Planning how to configure 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 1108 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 128 on page 1114, 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 1115. You also have to manage the switch from one transmission queue to the other.

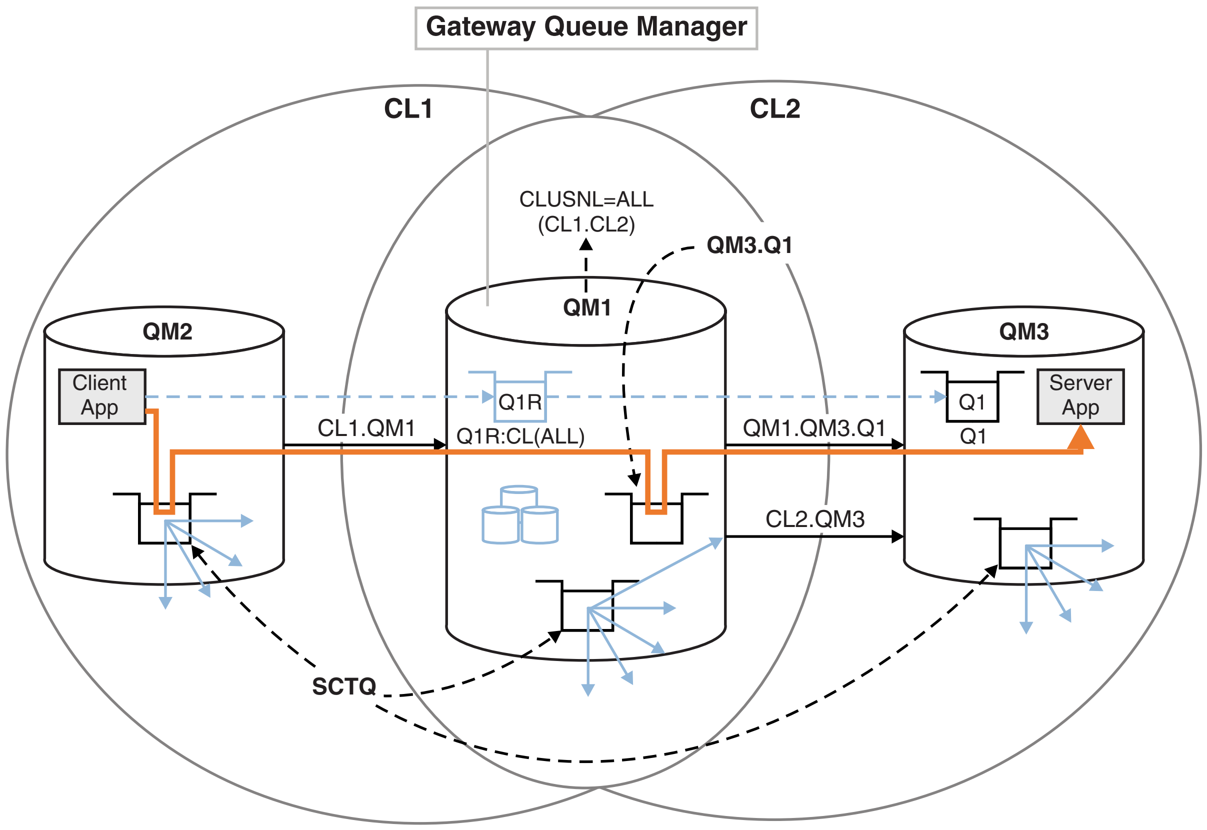


Figure 128. Client-server application deployed to hub and spoke cluster architecture using remote queue definitions

Procedure

1. Create a channel to separate the message traffic for Q1 from the gateway queue manager
   a. Create a sender channel on the gateway queue manager, QM1, to the target queue manager, QM3.
   ```
   DEFINE CHANNEL(QM1.QM3.Q1) CHLTYPE(SDR) CONNAME(QM3HostName(1413)) XMITQ(QM3.Q1) REPLACE
   ```
   b. Create a receiver channel on the target queue manager, QM3.
   ```
   DEFINE CHANNEL(QM1.QM3.Q1) CHLTYPE(RCVR) REPLACE
   ```
2. Create a transmission queue on the gateway queue manager for message traffic to Q1
   ```
   DEFINE QLOCAL(QM3.Q1) USAGE(XMITQ) REPLACE
   START CHANNEL(QM1.QM3.Q1)
   ```
   Starting the channel that is associated with the transmission queue, associates the transmission queue with the channel. The channel starts automatically, once the transmission queue has been associated with the channel.
3. Supplement the clustered queue alias definition for Q1 on the gateway queue manager with a clustered remote queue definition.
   ```
   DEFINE QREMOTE CLUSNL(ALL) RNAME(Q1) RQMNAME(QM3) XMITQ(QM3.Q1) REPLACE
   ```

What to do next

Test the configuration by sending a message to Q1 on QM3 from QM2 using the clustered queue remote definition Q1R on the gateway queue manager QM1.

1. Run the sample program **amqsput** on QM2 to put a message.
   ```
   C:\IBM\MQ>amqsput Q1R QM2
   Sample AMQSPUT0 start
   ```

```
target queue is Q1R
Sample request message from QM2 to Q1 using Q1R

Sample AMQSPUT0 end
```

2. Run the sample program **amqsget** to get the message from Q1 on QM3

```
C:\IBM\MQ>amqsget Q1 QM3
Sample AMQSGET0 start
message <Sample request message from QM2 to Q1 using Q1R>
no more messages
Sample AMQSGET0 end
```

Related tasks:

"Adding a queue manager to a cluster: separate transmission queues" on page 1087
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 information:

Clustering: Application isolation using multiple cluster transmission queues

Clustering: Planning how to configure cluster transmission queues

Access control and multiple cluster transmission queues

Adding a cluster transmit queue to isolate cluster message traffic sent from a gateway queue manager:

Modify the configuration of overlapping clusters that use a gateway queue manager. After the modification messages are transferred to an application from the gateway queue manager without using the same transmission queue or channels as other cluster messages. The solution uses an additional cluster transmission queue to separate message traffic to a single queue manager in a cluster.

Before you begin

1. The gateway queue manager must be on Version 7.5, or later.
2. Construct the overlapping clusters shown in Client-server application deployed to hub and spoke architecture using IBM MQ clusters in "Creating two-overlapping clusters with a gateway queue manager" on page 1108 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 129 on page 1116.

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 1113:
- It requires fewer additional definitions.
- It supports workload balancing between multiple copies of the target queue, Q1, on different queue managers in the same cluster, CL2.
- The gateway queue manager switches automatically to the new configuration when the channel restarts without loosing any messages.
- The gateway queue manager continues to forward messages in the same order as it received them. It does so, even if the switch takes place with messages for the queue Q1 at QM3 still on SYSTEM.CLUSTER.TRANSMIT.QUEUE.

The configuration to isolate cluster message traffic in Figure 129 on page 1116 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 1113. 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 129 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 1118.

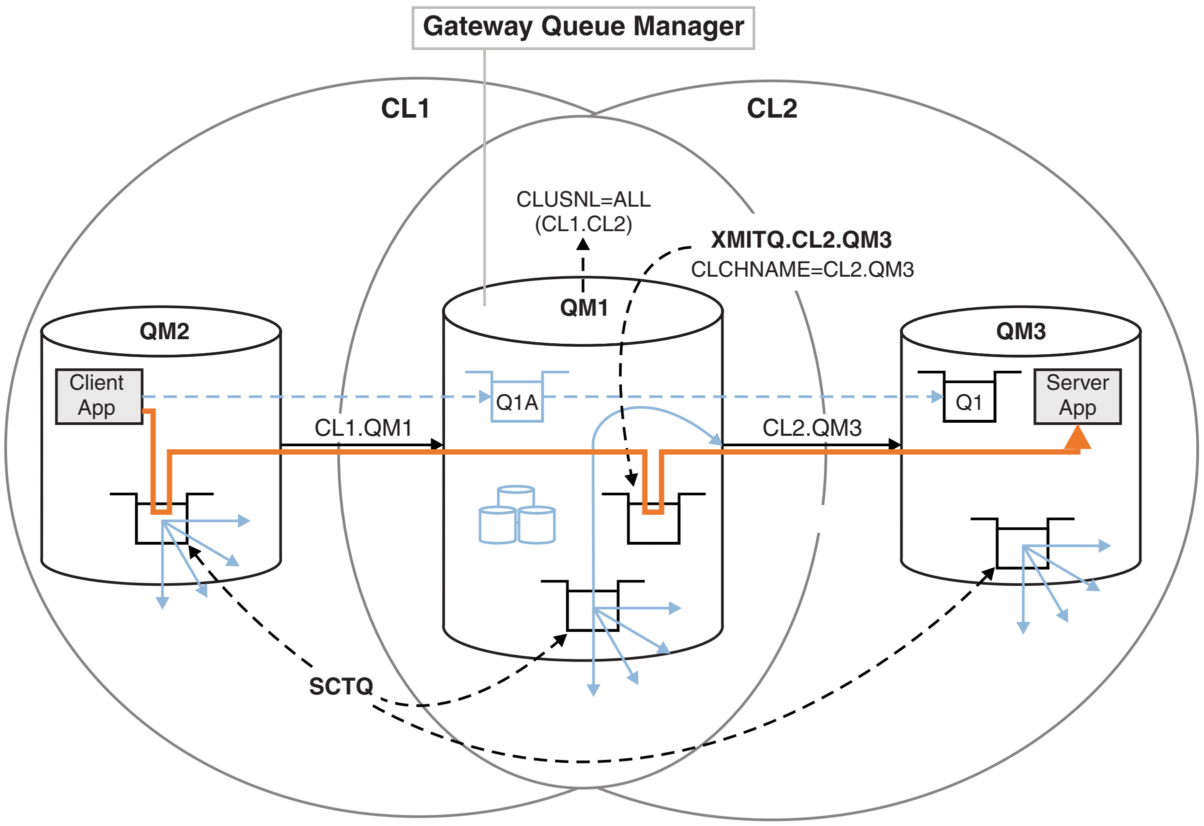


Figure 129. Client-server application deployed to hub and spoke architecture using an additional cluster transmission queue.

Procedure

1. Create an additional cluster transmission queue for the cluster-sender channel CL2.QM3 on the gateway queue manager, QM1.

```
 *... on QM1
DEFINE QLOCAL(XMITQ.CL2.QM3) USAGE(XMITQ) CLCHNAME(CL2.QM3)
```

2. Switch to using the transmission queue, XMITQ.CL2.QM3.
   a. Stop the cluster-sender channel CL2.QM3.

   ```
   *... On QM1
   STOP CHANNEL(CL2.QM3)
   ```

   The response is that the command is accepted:

   ```
   AMQ8019: Stop IBM MQ channel accepted.
   ```

   b. Check that the channel CL2.QM3 is stopped

If the channel does not stop, you can run the **STOP CHANNEL** command again with the FORCE option. An example of setting the FORCE option would be if the channel does not stop, and you cannot restart the other queue manager to synchronize the channel.

```
*... On QM1
start
```

The response is a summary of the channel status

```
AMQ8417: Display Channel Status details.
CHANNEL(CL2.QM3)             CHLTYPE(CLUSSDR)
CONNAME(127.0.0.1(1413))     CURRENT
RQMNAME(QM3)                 STATUS(STOPPED)
SUBSTATE(MQGET)              XMITQ(SYSTEM.CLUSTER.TRANSMIT.QUEUE)
```

c. Start the channel, CL2.QM3.

```
*... On QM1
START CHANNEL(CL2.QM3)
```

The response is that the command is accepted:

```
AMQ8018: Start IBM MQ channel accepted.
```

d. Check the channel started.

```
*... On QM1
DISPLAY CHSTATUS(CL2.QM3)
```

The response is a summary of the channel status:

```
AMQ8417: Display Channel Status details.
CHANNEL(CL2.QM3)                  CHLTYPE(CLUSSDR)
CONNAME(127.0.0.1(1413))          CURRENT
RQMNAME(QM3)                      STATUS(RUNNING)
SUBSTATE(MQGET)                   XMITQ(XMITQ.CL2.QM3)
```

e. Check the transmission queue was switched.

Monitor the gateway queue manager error log for the message " AMQ7341 The transmission queue for channel CL2.QM3 is XMITQ.CL2.QM3 ".

What to do next

Test the separate transmission queue by sending a message from QM2 to Q1 on QM3 using the queue alias definition Q1A

1. Run the sample program **amqsput** on QM2 to put a message.

```
C:\IBM\MQ>amqsput Q1A QM2
Sample AMQSPUT0 start
target queue is Q1A
Sample request message from QM2 to Q1 using Q1A

Sample AMQSPUT0 end
```

2. Run the sample program **amqsget** to get the message from Q1 on QM3

```
C:\IBM\MQ>amqsget Q1 QM3
Sample AMQSGET0 start
message <Sample request message from QM2 to Q1 using Q1A>
no more messages
Sample AMQSGET0 end
```

Related concepts:

"Working with cluster transmission queues and cluster-sender channels" on page 1068
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 queue manager to a cluster: separate transmission queues" on page 1087
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 information:

Access control and multiple cluster transmission queues

Clustering: Application isolation using multiple cluster transmission queues

Clustering: Planning how to configure 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 129 on page 1116.

1. The gateway queue manager must be on Version 7.5, or later.
2. Construct the overlapping clusters shown in Client-server application deployed to hub and spoke architecture using IBM MQ clusters in "Creating two-overlapping clusters with a gateway queue manager" on page 1108 by following the steps in that task.
3. Do the steps in Figure 129 on page 1116 in "Adding a cluster transmit queue to isolate cluster message traffic sent from a gateway queue manager" on page 1115 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 1115 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 1115 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 130 on page 1119. 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 1115 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.

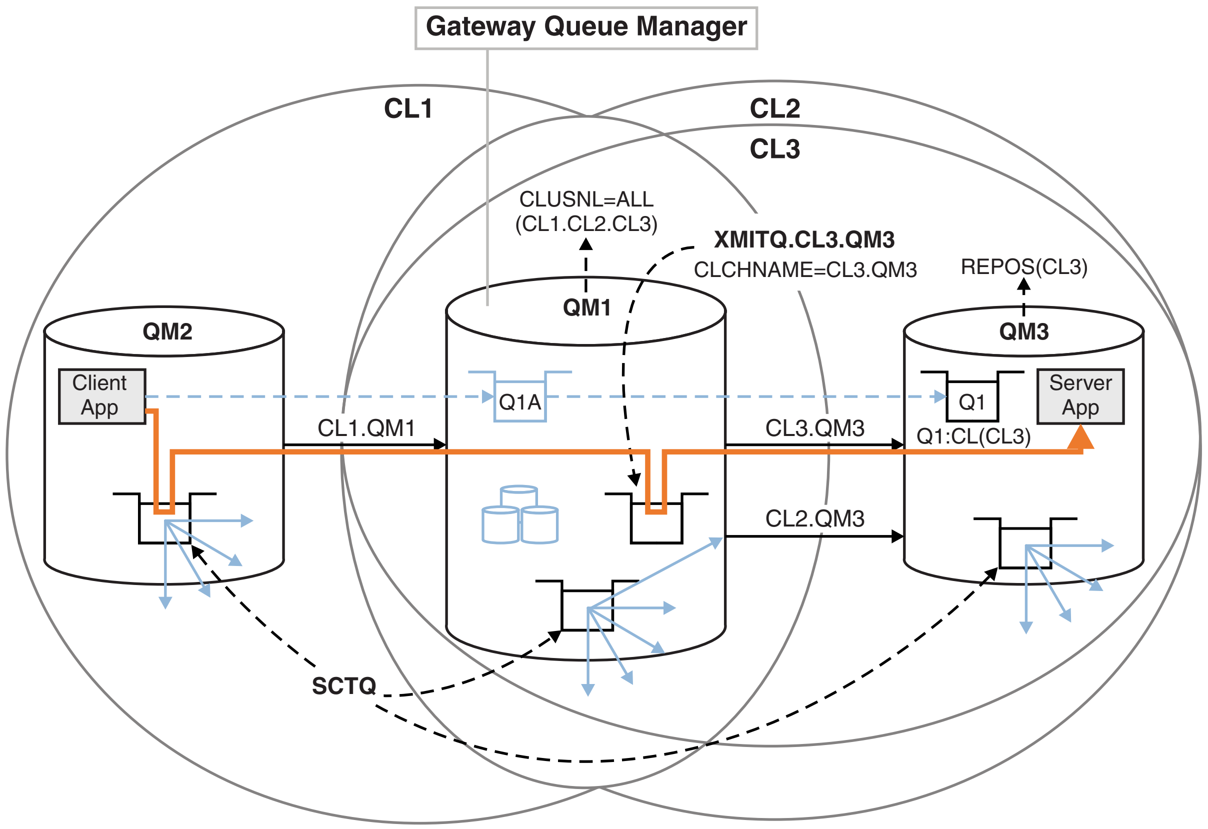


Figure 130. Using an additional cluster to separate message traffic in the gateway queue manager that goes to one of a number of cluster queues on the same queue manager

Procedure

1. Alter the queue managers QM3 and QM5 to make them repositories for both CL2 and CL3.

   To make a queue manager a member of multiple clusters, it must use a cluster name list to identify the clusters it is a member of.

   ```
   *... On QM3 and QM5
   DEFINE NAMELIST(CL23) NAMES(CL2, CL3) REPLACE
   ALTER QMGR REPOS(' ') REPOSNL(CL23)
   ```

2. Define the channels between the queue managers QM3 and QM5 for CL3.

   ```
   *... On QM3
   DEFINE CHANNEL(CL3.QM5) CHLTYPE(CLUSSDR) CONNAME('localhost(1415)') CLUSTER(CL3) REPLACE
   DEFINE CHANNEL(CL3.QM3) CHLTYPE(CLUSRCVR) CONNAME('localhost(1413)') CLUSTER(CL3) REPLACE
   ```

```
*... On QM5
DEFINE CHANNEL(CL3.QM3) CHLTYPE(CLUSSDR) CONNAME('localhost(1413)') CLUSTER(CL3) REPLACE
DEFINE CHANNEL(CL3.QM5) CHLTYPE(CLUSRCVR) CONNAME('localhost(1415)') CLUSTER(CL3) REPLACE
```

3. Add the gateway queue manager to CL3.

   Add the gateway queue manager by adding QM1 to CL3 as a partial repository. Create a partial repository by adding cluster-sender and cluster-receiver channels to QM1.

   Also, add CL3 to the name list of all clusters connected to the gateway queue manager.

   ```
   *... On QM1
   DEFINE CHANNEL(CL3.QM3) CHLTYPE(CLUSSDR) CONNAME('localhost(1413)') CLUSTER(CL3) REPLACE
   DEFINE CHANNEL(CL3.QM1) CHLTYPE(CLUSRCVR) CONNAME('localhost(1411)') CLUSTER(CL3) REPLACE
   ALTER NAMELIST(ALL) NAMES(CL1, CL2, CL3)
   ```

4. Add a cluster transmission queue to the gateway queue manager, QM1, for messages going to CL3 on QM3.

   Initially, stop the cluster-sender channel transferring messages from the transmission queue until you are ready to switch transmission queues.

   ```
   *... On QM1
   DEFINE QLOCAL(XMITQ.CL3.QM3) USAGE(XMITQ) CLCHNAME(CL3.QM3) GET(DISABLED) REPLACE
   ```

5. Drain messages from the existing cluster transmission queue XMITQ.CL2.QM3.

   This subprocedure is intended to preserve the order of messages in Q1 to match the order they arrived at the gateway queue manager. With clusters, message ordering is not fully guaranteed, but is likely. If guaranteed message ordering is required, applications must define the order of messages; see The order in which messages are retrieved from a queue.

   a. Change the target queue Q1 on QM3 from CL2 to CL3.

      ```
      *... On QM3
      ALTER QLOCAL(Q1) CLUSTER(CL3)
      ```

   b. Monitor XMITQ.CL3.QM3 until messages start to be delivered to it.

      Messages start to be delivered to XMITQ.CL3.QM3 when the switch of Q1 to CL3 is propagated to the gateway queue manager.

      ```
      *... On QM1
      DISPLAY QUEUE(XMITQ.CL3.QM3) CURDEPTH
      ```

   c. Monitor XMITQ.CL2.QM3 until it has no messages waiting to be delivered to Q1 on QM3.

      Note: XMITQ.CL2.QM3 might be storing messages for other queues on QM3 that are members of CL2, in which case the depth might not go to zero.

      ```
      *... On QM1
      DISPLAY QUEUE(XMITQ.CL2.QM3) CURDEPTH
      ```

   d. Enable get from the new cluster transmission queue, XMITQ.CL3.QM3

      ```
      *... On QM1
      ALTER QLOCAL(XMITQ.CL3.QM3) GET(ENABLED)
      ```

6. Remove the old cluster transmission queue, XMITQ.CL2.QM3, if it is no longer required.

   Messages for cluster queues in CL2 on QM3 revert to using the default cluster transmission queue on the gateway queue manager, QM1. The default cluster transmission queue is either SYSTEM.CLUSTER.TRANSMIT.QUEUE, or SYSTEM.CLUSTER.TRANSMIT.CL2.QM3. Which one depends on whether the value of the queue manager attribute **DEFCLXQ** on QM1 is SCTQ or CHANNEL. The queue manager transfers messages from XMITQ.CL2.QM3 automatically when the cluster-sender channel CL2.QM3 next starts.

   a. Change the transmission queue, XMITQ.CL2.QM3, from being a cluster transmission queue to being a normal transmission queue.

      This breaks the association of the transmission queue with any cluster-sender channels. In response, IBM MQ automatically transfers messages from XMITQ.CL2.QM3 to the default cluster transmission queue when the cluster-sender channel is next started. Until then, messages for CL2 on QM3 continue to be placed on XMITQ.CL2.QM3.

```
*... On QM1
ALTER QLOCAL(XMITQ.CL2.QM3) CLCHNAME(' ')
```

b. Stop the cluster-sender channel CL2.QM3.

   Stopping and restarting the cluster-sender channel initiates the transfer of messages from XMITQ.CL2.QM3 to the default cluster transmission queue. Typically you would stop and start the channel manually to start the transfer. The transfer starts automatically if the channel restarts after shutting down on the expiry of its disconnect interval.

```
*... On QM1
STOP CHANNEL(CL2.QM3)
```

   The response is that the command is accepted:

```
AMQ8019: Stop IBM MQ channel accepted.
```

c. Check that the channel CL2.QM3 is stopped

   If the channel does not stop, you can run the **STOP CHANNEL** command again with the FORCE option. An example of setting the FORCE option would be if the channel does not stop, and you cannot restart the other queue manager to synchronize the channel.

```
*... On QM1
DISPLAY CHSTATUS(CL2.QM3)
```

   The response is a summary of the channel status

```
AMQ8417: Display Channel Status details.
CHANNEL(CL2.QM3)                    CHLTYPE(CLUSSDR)
CONNAME(127.0.0.1(1413))            CURRENT
RQMNAME(QM3)                        STATUS(STOPPED)
SUBSTATE(MQGET)                     XMITQ(XMITQ.CL2.QM3)
```

d. Start the channel, CL2.QM3.

```
*... On QM1
START CHANNEL(CL2.QM3)
```

   The response is that the command is accepted:

```
AMQ8018: Start IBM MQ channel accepted.
```

e. Check the channel started.

```
*... On QM1
DISPLAY CHSTATUS(CL2.QM3)
```

   The response is a summary of the channel status:

```
AMQ8417: Display Channel Status details.
CHANNEL(CL2.QM3)      CHLTYPE(CLUSSDR)
CONNAME(127.0.0.1(1413))  CURRENT
RQMNAME(QM3)        STATUS(RUNNING)
SUBSTATE(MQGET)      XMITQ(SYSTEM.CLUSTER.TRANSMIT. QUEUE|CL2.QM3)
```

f. Monitor the gateway queue manager error log for the message " AMQ7341 `The transmission queue for channel CL2.QM3 is SYSTEM.CLUSTER.TRANSMIT.` *QUEUE|CL2.QM3* ".

g. Delete the cluster transmission queue, XMITQ.CL2.QM3.

```
*... On QM1
DELETE QLOCAL(XMITQ.CL2.QM3)
```

What to do next

Test the separately clustered queue by sending a message from QM2 to Q1 on QM3 using the queue alias definition Q1A

1. Run the sample program **amqsput** on QM2 to put a message.

```
C:\IBM\MQ>amqsput Q1A QM2
Sample AMQSPUT0 start
target queue is Q1A
Sample request message from QM2 to Q1 using Q1A

Sample AMQSPUT0 end
```

2. Run the sample program **amqsget** to get the message from Q1 on QM3

```
C:\IBM\MQ>amqsget Q1 QM3
Sample AMQSGET0 start
message <Sample request message from QM2 to Q1 using Q1A>
no more messages
Sample AMQSGET0 end
```

Related concepts:

"Working with cluster transmission queues and cluster-sender channels" on page 1068
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 queue manager to a cluster: separate transmission queues" on page 1087
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 information:

Access control and multiple cluster transmission queues

Clustering: Application isolation using multiple cluster transmission queues

Clustering: Planning how to configure cluster transmission queues

Changing the default to separate cluster transmission queues to isolate message traffic:

You can change the default way a queue manager stores messages for a clustered queue or topic on a transmission queue. Changing the default provides you with a way to isolate cluster messages on a gateway queue manager.

Before you begin

1. The gateway queue manager must be on Version 7.5, or later.
2. Construct the overlapping clusters shown in Client-server application deployed to hub and spoke architecture using IBM MQ clusters in "Creating two-overlapping clusters with a gateway queue manager" on page 1108 by following the steps in that task.

About this task

To implement the architecture with multiple clusters queue, your gateway queue manager must be on Version 7.5, or later. All you do to use multiple cluster transmission queues is to change the default cluster transmission queue type on the gateway queue manager. Change the value of the queue manager attribute **DEFCLXQ** on QM1 from SCTQ to CHANNEL ; see Figure 131 on page 1123. 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 1087.

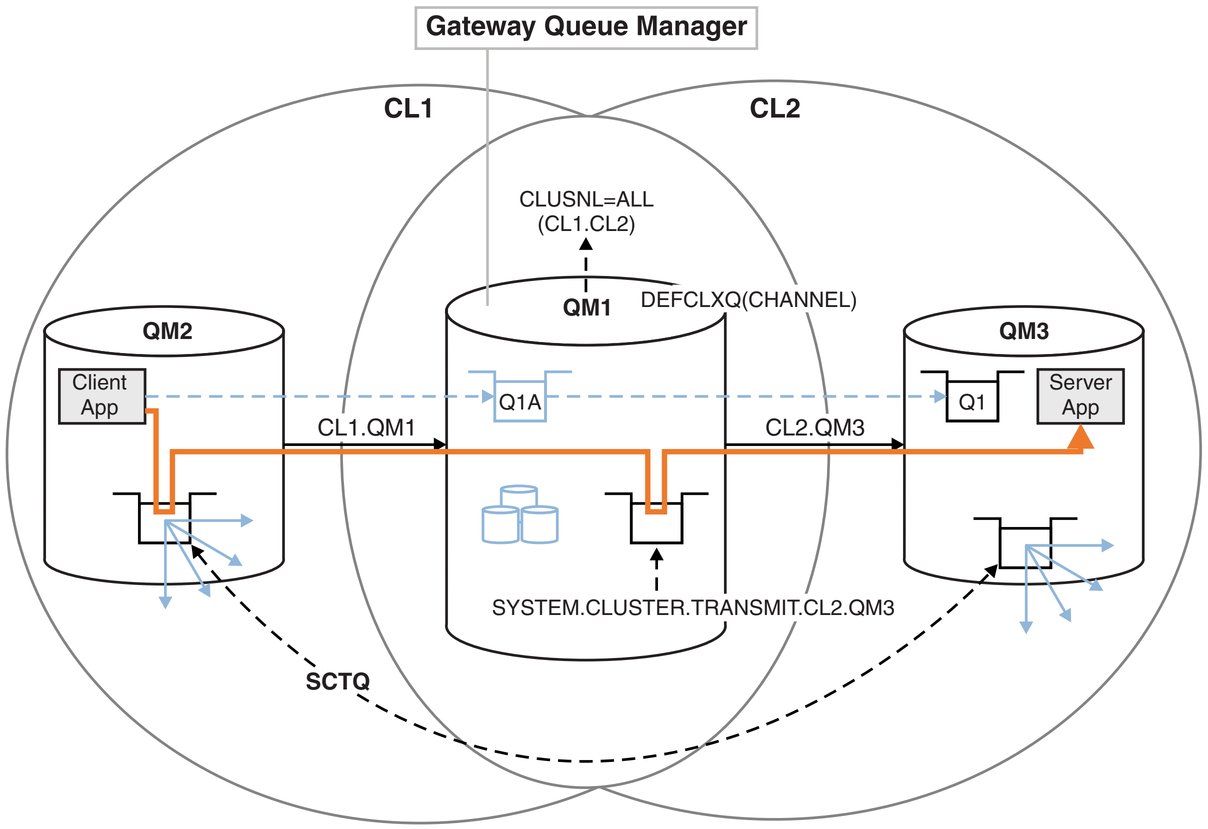


Figure 131. Client-server application deployed to hub and spoke architecture with separate cluster transmission queues on the gateway queue manager.

Procedure

1. Change the gateway queue manager to use separate cluster transmission queues.

```
*... On QM1
ALTER QMGR DEFCLXQ(CHANNEL)
```

2. Switch to the separate cluster transmission queues.

   Any cluster-sender channel that is not running switches to using separate cluster transmission queues when it next starts.

   To switch the running channels, either restart the queue manager, or follow these steps:

   a. List the cluster-sender channels that are running with SYSTEM.CLUSTER.TRANSMIT.QUEUE.

```
*... On QM1
DISPLAY CHSTATUS(*) WHERE(XMITQ EQ 'SYSTEM.CLUSTER.TRANSMIT.QUEUE')
```

   The response is list of channel status reports:

```
AMQ8417: Display Channel Status details.
CHANNEL(CL1.QM2)              CHLTYPE(CLUSSDR)
CONNAME(127.0.0.1(1412))      CURRENT
RQMNAME(QM2)                  STATUS(RUNNING)
SUBSTATE(MQGET)               XMITQ(SYSTEM.CLUSTER.TRANSMIT.QUEUE)
AMQ8417: Display Channel Status details.
CHANNEL(CL2.QM3)              CHLTYPE(CLUSSDR)
CONNAME(127.0.0.1(1413))      CURRENT
RQMNAME(QM3)                  STATUS(RUNNING)
```

```
SUBSTATE(MQGET)             XMITQ(SYSTEM.CLUSTER.TRANSMIT.QUEUE)
AMQ8417: Display Channel Status details.
CHANNEL(CL2.QM5)            CHLTYPE(CLUSSDR)
CONNAME(127.0.0.1(1415))    CURRENT
RQMNAME(QM5)                STATUS(RUNNING)
SUBSTATE(MQGET)             XMITQ(SYSTEM.CLUSTER.TRANSMIT.QUEUE)
AMQ8417: Display Channel Status details.
CHANNEL(CL1.QM4)            CHLTYPE(CLUSSDR)
CONNAME(127.0.0.1(1414))    CURRENT
RQMNAME(QM4)                STATUS(RUNNING)
SUBSTATE(MQGET)             XMITQ(SYSTEM.CLUSTER.TRANSMIT.QUEUE)
```

b. Stop the channels that are running

   For each channel in the list, run the command:

```
*... On QM1
STOP CHANNEL(ChannelName)
```

   Where *ChannelName* is each of `CL1.QM2`, `CL1.QM4`, `CL1.QM3`, `CL1.QM5`.

   The response is that the command is accepted:

```
AMQ8019: Stop IBM MQ channel accepted.
```

c. Monitor which channels are stopped

```
*... On QM1
DISPLAY CHSTATUS(*) WHERE(XMITQ EQ 'SYSTEM.CLUSTER.TRANSMIT.QUEUE')
```

   The response is a list of channels that are still running and channels that are stopped:

```
AMQ8417: Display Channel Status details.
CHANNEL(CL1.QM2)           CHLTYPE(CLUSSDR)
CONNAME(127.0.0.1(1412))   CURRENT
RQMNAME(QM2)               STATUS(STOPPED)
SUBSTATE( )                XMITQ(SYSTEM.CLUSTER.TRANSMIT.QUEUE)
AMQ8417: Display Channel Status details.
CHANNEL(CL2.QM3)           CHLTYPE(CLUSSDR)
CONNAME(127.0.0.1(1413))   CURRENT
RQMNAME(QM3)               STATUS(STOPPED)
SUBSTATE( )                XMITQ(SYSTEM.CLUSTER.TRANSMIT.QUEUE)
AMQ8417: Display Channel Status details.
CHANNEL(CL2.QM5)           CHLTYPE(CLUSSDR)
CONNAME(127.0.0.1(1415))   CURRENT
RQMNAME(QM5)               STATUS(STOPPED)
SUBSTATE( )                XMITQ(SYSTEM.CLUSTER.TRANSMIT.QUEUE)
AMQ8417: Display Channel Status details.
CHANNEL(CL1.QM4)           CHLTYPE(CLUSSDR)
CONNAME(127.0.0.1(1414))   CURRENT
RQMNAME(QM4)               STATUS(STOPPED)
SUBSTATE( )                XMITQ(SYSTEM.CLUSTER.TRANSMIT.QUEUE)
```

d. Start each stopped channel.

   Do this step for all the channels that were running. If a channel does not stop, you can run the **STOP CHANNEL** command again with the `FORCE` option. An example of setting the `FORCE` option would be if the channel does not stop, and you cannot restart the other queue manager to synchronize the channel.

```
*... On QM1
START CHANNEL(CL2.QM5)
```

   The response is that the command is accepted:

```
AMQ8018: Start IBM MQ channel accepted.
```

e. Monitor the transmission queues being switched.

   Monitor the gateway queue manager error log for the message “ AMQ7341 The transmission queue for channel CL2.QM3 is SYSTEM.CLUSTER.TRANSMIT. *QUEUE|CL2.QM3* ”.

f. Check that SYSTEM.CLUSTER.TRANSMIT.QUEUE is no longer used

```
*... On QM1
DISPLAY CHSTATUS(*) WHERE(XMITQ EQ 'SYSTEM.CLUSTER.TRANSMIT.QUEUE')
DISPLAY QUEUE(SYSTEM.CLUSTER.TRANSMIT.QUEUE) CURDEPTH
```

   The response is list of channel status reports, and the depth of SYSTEM.CLUSTER.TRANSMIT.QUEUE:

```
AMQ8420: Channel Status not found.
AMQ8409: Display Queue details.
QUEUE(SYSTEM.CLUSTER.TRANSMIT.QUEUE)    TYPE(QLOCAL)
CURDEPTH(0)
```

g. Monitor which channels are started

```
*... On QM1
DISPLAY CHSTATUS(*) WHERE(XMITQ LK 'SYSTEM.CLUSTER.TRANSMIT.*')
```

   The response is a list of the channels, in this case already running with the new default cluster transmission queues:

```
AMQ8417: Display Channel Status details.
CHANNEL(CL1.QM2)                        CHLTYPE(CLUSSDR)
CONNAME(127.0.0.1(1412))                CURRENT
RQMNAME(QM2)                            STATUS(RUNNING)
SUBSTATE(MQGET)
XMITQ(SYSTEM.CLUSTER.TRANSMIT.CL1.QM2)
AMQ8417: Display Channel Status details.
CHANNEL(CL2.QM3)                        CHLTYPE(CLUSSDR)
CONNAME(127.0.0.1(1413))                CURRENT
RQMNAME(QM3)                            STATUS(RUNNING)
SUBSTATE(MQGET)
XMITQ(SYSTEM.CLUSTER.TRANSMIT.CL2.QM3)
AMQ8417: Display Channel Status details.
CHANNEL(CL2.QM5)                        CHLTYPE(CLUSSDR)
CONNAME(127.0.0.1(1415))                CURRENT
RQMNAME(QM5)                            STATUS(RUNNING)
SUBSTATE(MQGET)
XMITQ(SYSTEM.CLUSTER.TRANSMIT.CL2.QM5)
AMQ8417: Display Channel Status details.
CHANNEL(CL1.QM4)                        CHLTYPE(CLUSSDR)
CONNAME(127.0.0.1(1414))                CURRENT
RQMNAME(QM4)                            STATUS(RUNNING)
SUBSTATE(MQGET)
XMITQ(SYSTEM.CLUSTER.TRANSMIT.CL1.QM4)
```

What to do next

1. Test the automatically defined cluster transmission queue by sending a message from QM2 to Q1 on QM3, resolving queue name with the queue alias definition Q1A

   a. Run the sample program **amqsput** on QM2 to put a message.

```
C:\IBM\MQ>amqsput Q1A QM2
Sample AMQSPUT0 start
target queue is Q1A
```

```
Sample request message from QM2 to Q1 using Q1A

Sample AMQSPUT0 end
```

b. Run the sample program **amqsget** to get the message from Q1 on QM3

```
C:\IBM\MQ>amqsget Q1 QM3
Sample AMQSGET0 start
message <Sample request message from QM2 to Q1 using Q1A>
no more messages
Sample AMQSGET0 end
```

2. Consider whether to reconfigure security, by configuring security for the cluster queues on the queue managers where messages for the cluster queues originate.

Related tasks:

"Adding a queue manager to a cluster: separate transmission queues" on page 1087
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 information:

Access control and multiple cluster transmission queues

Clustering: Application isolation using multiple cluster transmission queues

Clustering: Planning how to configure 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 1092. 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 queuesare 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

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 1130.

Before you remove the queue manager from the cluster, you must 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 1096. 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 1126.
- 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 1172.

  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.

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 1085, and modified as described in “Removing a cluster queue from a queue manager” on page 1126.

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. Alter the manually defined cluster receiver channels to remove them from the cluster, on queue manager `LONDON`:
   ```
   ALTER CHANNEL(INVENTORY.LONDON) CHLTYPE(CLUSRCVR) CLUSTER(' ')
   ```
2. 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.
3. Monitor the cluster transmit queue, on queue manager `LONDON`, until there are no messages that are waiting to flow to any full repository in the cluster.
   ```
   DISPLAY CHSTATUS(INVENTORY.LONDON) XQMSGSA
   ```
   If messages remain on the transmit queue, determine why they are not being sent to the `PARIS` and `NEWYORK` full repositories before continuing.

Results

The queue manager `LONDON` is no longer part of the cluster. However, it can still function as an independent queue manager.

What to do next

The result of these changes can be confirmed by issuing the following command on the remaining members of the cluster:
```
DISPLAY CLUSQMGR(LONDON)
```

The queue manager continues to be displayed until the auto-defined cluster sender channels to it have stopped. You can wait for this to happen, or, continue to monitor for active instances by issuing the following command:
```
DISPLAY CHANNEL(INVENTORY.LONDON)
```

When you are confident that no more messages are being delivered to this queue manager, you can stop the cluster sender channels to `LONDON` by issuing the following command on the remaining members of the cluster:
```
STOP CHANNEL(INVENTORY.LONDON) STATUS(INACTIVE)
```

After the changes are propagated throughout the cluster, and no more messages are being delivered to this queue manager, stop and delete the `CLUSRCVR` channel on `LONDON`:

```
STOP CHANNEL(INVENTORY.LONDON)
DELETE CHANNEL(INVENTORY.LONDON)
```

The removed queue manager can be added back into the cluster at a later point as described in "Adding a queue manager to a cluster" on page 1085. 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 1132.

Removing a queue manager from a cluster: Alternative method:

Remove a queue manager from a cluster, in scenarios where, because of a significant system or configuration issue, the queue manager cannot communicate with any full repository in the cluster.

Before you begin

This alternative method of removing a queue manager from a cluster manually stops and deletes all cluster channels linking the removed queue manager to the cluster, and forcibly removes the queue manager from the cluster. This method is used in scenarios where the queue manager that is being removed cannot communicate with any of the full repositories. This might be (for example) because the queue manager has stopped working, or because there has been a prolonged communications failure between the queue manager and the cluster. Otherwise, use the most common method: "Removing a queue manager from a cluster" on page 1128.

Before you remove the queue manager from the cluster, you must 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 1096. 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 1126.
- 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 1172.

  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.

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 1085, and modified as described in "Removing a cluster queue from a queue manager" on page 1126.

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. 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)
   ```
2. 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.
3. Delete the manually defined cluster channels, on queue manager `LONDON`:
   ```
   DELETE CHANNEL(INVENTORY.NEWYORK)
   DELETE CHANNEL(INVENTORY.TORONTO)
   ```
4. 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 1085. 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.”

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.

Important: If you remove a queue manager from a cluster and refresh it using REPOS(YES), you will not be able to add it back in again. Even refreshing the other queue managers in the cluster will not enable it to be added back in. Only waiting for the cluster cache expiry time to expire will enable it to be added back in.

About this task

When a queue manager is removed from a cluster, it retains knowledge of the remaining cluster members for up to 90 days. This can have system benefits, particularly if the queue manager quickly rejoins the cluster. When this knowledge eventually expires, it is deleted automatically. However, there are reasons why you might prefer to delete this information manually. For example:

- You might want to confirm that you have stopped every application on this queue manager that previously used cluster resources. Until the knowledge of the remaining cluster members expires, any such application continues to write to a transmit queue. After the cluster knowledge is deleted, the system generates an error message when such an application tries to use cluster resources.
- When you display status information for the queue manager, you might prefer not to see expiring information about remaining cluster members.

This task uses the `INVENTORY` cluster as an example. The `LONDON` queue manager has been removed from the `INVENTORY` cluster as described in “Removing a queue manager from a cluster” on page 1128. 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 information:

Clusters

Comparison of clustering and distributed queuing

Cluster components

Maintaining a queue manager

Suspend and resume a queue manager from a cluster to perform maintenance.

About this task

From time to time, you might need to perform maintenance on a queue manager that is part of a cluster. For example, you might need to take backups of the data in its queues, or apply fixes to the software. If the queue manager hosts any queues, its activities must be suspended. When the maintenance is complete, its activities can be resumed.

Procedure

1. Suspend a queue manager, by issuing the `SUSPEND QMGR` **runmqsc** command:

   ```
   SUSPEND QMGR CLUSTER(SALES)
   ```

   The `SUSPEND` **runmqsc** command notifies the queue managers in the `SALES` cluster that this queue manager has been suspended.

   The purpose of the `SUSPEND QMGR` command is only to advise other queue managers to avoid sending messages to this queue manager if possible. It does not mean that the queue manager is disabled. Some messages that have to be handled by this queue manager are still sent to it, for example when this queue manager is the only host of a clustered queue.

   While the queue manager is suspended the workload management routines avoid sending messages to it. Messages that have to be handled by that queue manager include messages sent by the local queue manager.

   IBM MQ uses a workload balancing algorithm to determine which destinations are suitable, rather than selecting the local queue manager whenever possible.

   a. Enforce the suspension of a queue manager by using the `FORCE` option on the `SUSPEND QMGR` command:

      ```
      SUSPEND QMGR CLUSTER(SALES) MODE(FORCE)
      ```

      `MODE(FORCE)` forcibly stops all inbound channels from other queue managers in the cluster. If you do not specify `MODE(FORCE)`, the default `MODE(QUIESCE)` applies.

2. Do whatever maintenance tasks are necessary.
3. Resume the queue manager by issuing the `RESUME QMGR` **runmqsc** command:

   ```
   RESUME QMGR CLUSTER(SALES)
   ```

Results

The `RESUME` **runmqsc** command notifies the full repositories that the queue manager is available again. The full repository queue managers disseminate this information to other queue managers that have requested updates to information concerning this queue manager.

Maintaining the cluster transmission queue

Make every effort to keep cluster transmission queues available. They are essential to the performance of clusters. z/OS On z/OS, set the `INDXTYPE` of a cluster transmission queue to `CORRELID`.

Before you begin

- Make sure that the cluster transmission queue does not become full.
- Take care not to issue an `ALTER` **`runmqsc`** command to set it either get disabled or put disabled accidentally.
- Make sure that the medium the cluster transmission queue is stored on z/OS (for example z/OS page sets) does not become full.

z/OS

About this task

The following procedure is only applicable to z/OS. z/OS

Procedure

Set the `INDXTYPE` of the cluster transmission queue to `CORRELID`

Refreshing a cluster queue manager

You can remove auto-defined channels and auto-defined cluster objects from the local repository using the `REFRESH CLUSTER` command. No messages are lost.

Before you begin

You might be asked to use the command by your IBM Support Center. Do not use the command without careful consideration. For example, for large clusters use of the **`REFRESH CLUSTER`** command can be disruptive to the cluster while it is in progress, and again at 27 day intervals thereafter when the cluster objects automatically send status updates to all interested queue managers. See Clustering: Using REFRESH CLUSTER best practices.

About this task

A queue manager can make a fresh start in a cluster. In normal circumstances, you do not need to use the `REFRESH CLUSTER` command.

Procedure

Issue the `REFRESH CLUSTER` **`MQSC`** command from a queue manager to remove auto-defined cluster queue manager and queue objects from the local repository.
The command only removes objects that refer to other queue managers, it does not remove objects relating to the local queue manager. The command also removes auto-defined channels. It removes channels that do not have messages on the cluster transmission queue and are not attached to a full repository queue manager.

Results

Effectively, the `REFRESH CLUSTER` command allows a queue manager to be cold-started with respect to its full repository content. IBM MQ ensures that no data is lost from your queues.

Related information:
Clustering: Using REFRESH CLUSTER best practices

Recovering a cluster queue manager

Bring the cluster information about a queue manager up to date using the `REFRESH CLUSTER` **`runmqsc`** command. Follow this procedure after recovering a queue manager from a point-in-time backup.

Before you begin

You have restored a cluster queue manager from a point-in-time backup of a linear log.

About this task

To recover a queue manager in a cluster, restore the queue manager, and then bring the cluster information up to date using the `REFRESH CLUSTER` **`runmqsc`** command.

Note: For large clusters, using the **`REFRESH CLUSTER`** command can be disruptive to the cluster while it is in progress, and again at 27 day intervals thereafter when the cluster objects automatically send status updates to all interested queue managers. See Refreshing in a large cluster can affect performance and availability of the cluster.

Procedure

Issue the `REFRESH CLUSTER` command on the restored queue manager for all clusters in which the queue manager participates.

What to do next

There is no need to issue the `REFRESH CLUSTER` command on any other queue manager.

Related information:
Clustering: Using REFRESH CLUSTER best practices

Configuring cluster channels for availability

Follow good configuration practices to keep cluster channels running smoothly if there are intermittent network stoppages.

Before you begin

Clusters relieve you of the need to define channels, but you still need to maintain them. The same channel technology is used for communication between queue managers in a cluster as is used in distributed queuing. To understand about cluster channels, you need to be familiar with matters such as:

- How channels operate
- How to find their status
- How to use channel exits

About this task

You might want to give some special consideration to the following points:

Procedure

Consider the following points when configuring cluster channels

- Choose values for HBINT or KAINT on cluster-sender channels and cluster-receiver channels that do not burden the network with lots of heartbeat or keep alive flows. An interval less than about 10 seconds gives false failures, if your network sometimes slows down and introduces delays of this length.
- Set the BATCHHB value to reduce the window for causing a marooned message because it is indoubt on a failed channel. An indoubt batch on a failed channel is more likely to occur if the batch is given longer to fill. If the message traffic along the channel is sporadic with long periods of time between bursts of messages a failed batch is more likely.
- A problem arises if the cluster-sender end of a channel fails and then tries to restart before the heartbeat or keep alive has detected the failure. The channel-sender restart is rejected if the cluster-receiver end of the channel has remained active. To avoid the failure, arrange for the cluster-receiver channel to be terminated and restarted when a cluster-sender channel attempts to restart.

  z/OS **On IBM MQ for z/OS**

  Control the problem of the cluster-receiver end of the channel has remaining active using the `ADOPTMCA` and `ADOPTCHK` parameters on `ALTER QMGR`.

  Multi **On Multiplatforms**

  Control the problem of the cluster-receiver end of the channel has remaining active using the `AdoptNewMCA`, `AdoptNewMCATimeout`, and `AdoptNewMCACheck` attributes in the `qm.ini` file or the Windows NT Registry.

Routing messages to and from clusters

Use queue aliases, queue manager aliases, and remote queue definitions to connect clusters to external queue managers and other clusters.

For details on routing messages to and from clusters, see the following subtopics:

Related concepts:

"Queue manager aliases and clusters" on page 1145
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 1148
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 1148
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 1063
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 1074
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 information:

Clusters

Comparison of clustering and distributed queuing

Components of a cluster

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 132 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)
```

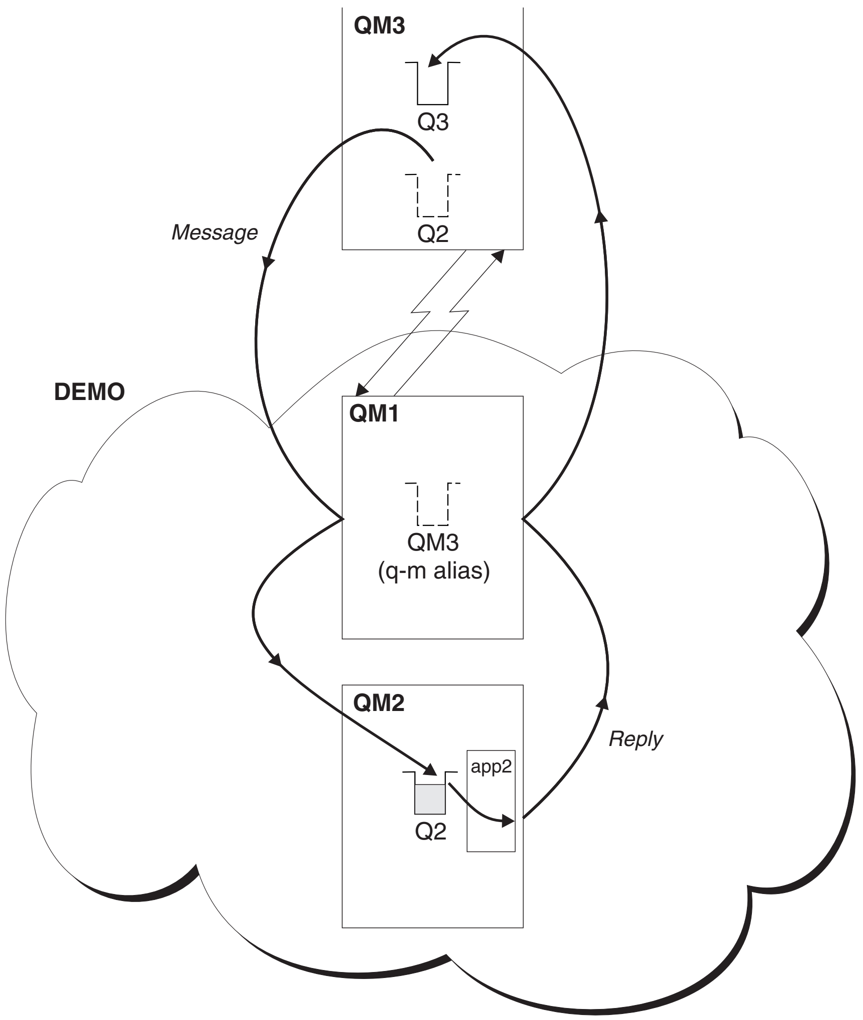


Figure 132. Putting from a queue manager outside the cluster

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 132 on page 1137.

   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.

      1) 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

2) 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 tasks:

"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.

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" 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 1137, to hide the name of the target queue manager inside the cluster.

Procedure

In the example, see Figure 133, define a queue manager alias on the gateway queue manager QM1 called DEMO:

```
DEFINE QREMOTE(DEMO) RNAME(' ') RQMNAME(' ')
```

The QREMOTE definition on QM1 makes the queue manager alias DEMO known to the gateway queue

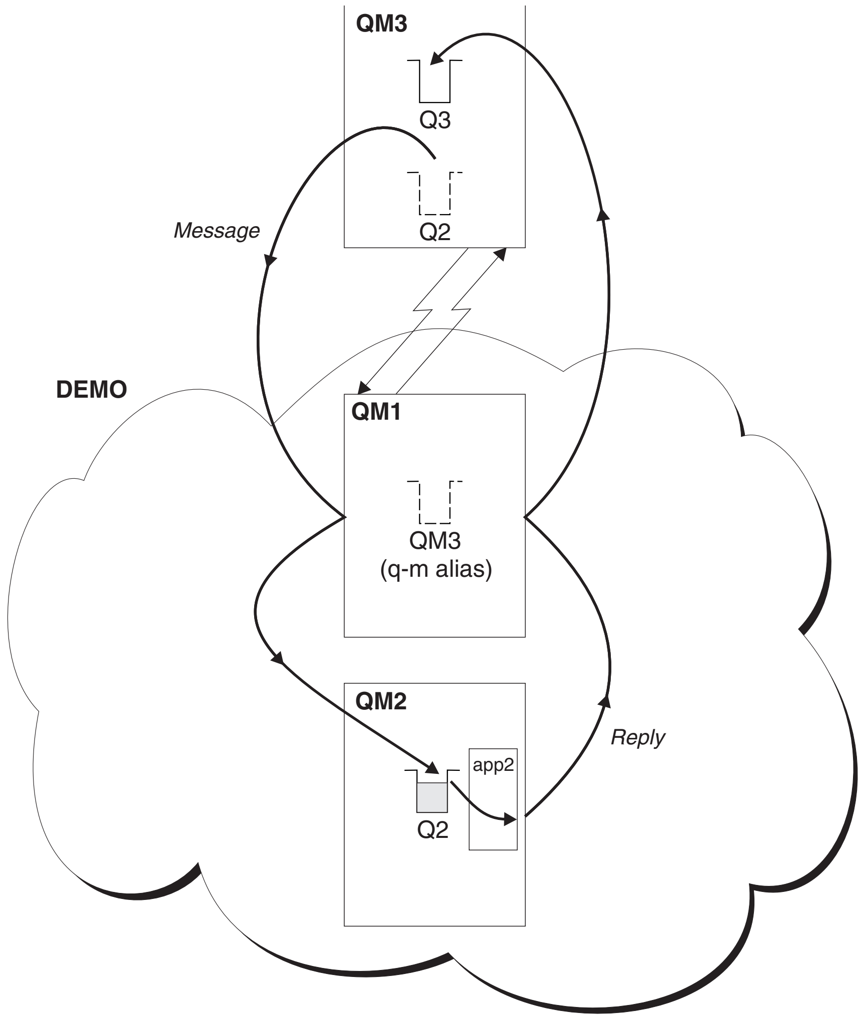


Figure 133. Putting from a queue manager outside the cluster

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 134 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.

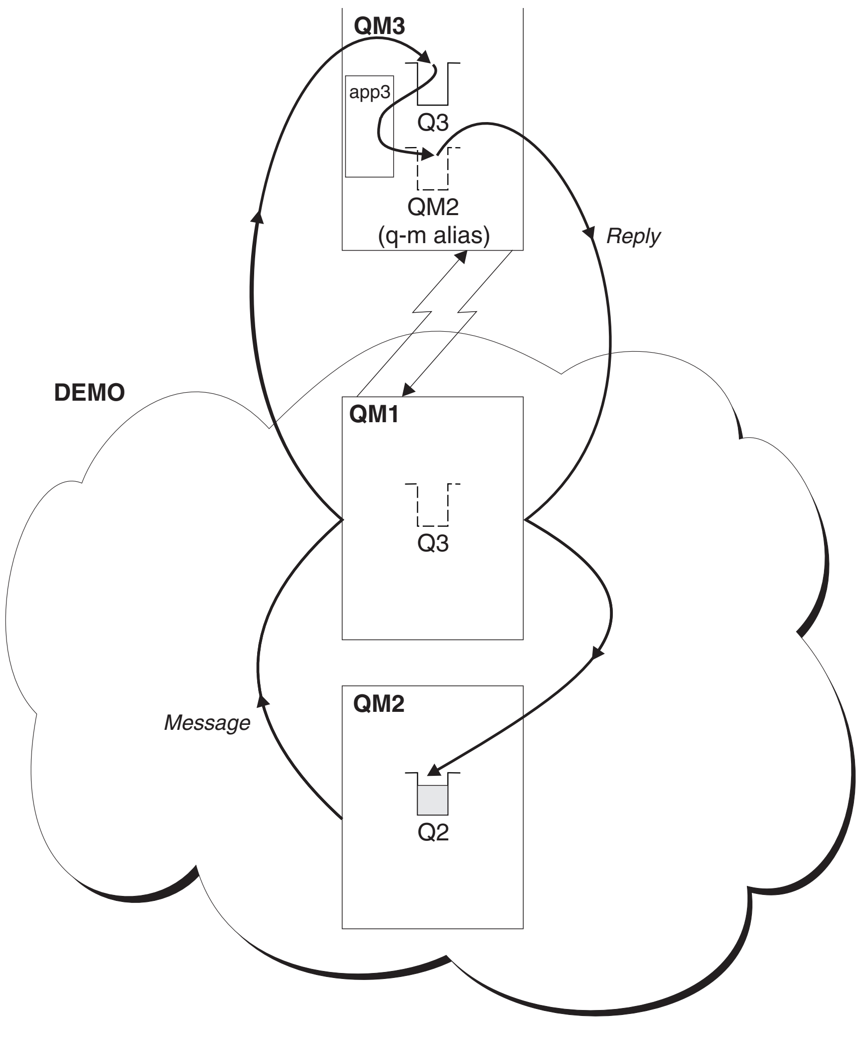


Figure 134. Putting to a queue manager outside the cluster

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 134 on page 1141.

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.

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 132 on page 1137 in "Configuring request/reply to a cluster" on page 1137.

About this task

In this scenario, the queue manager outside the cluster, QM3 in Figure 135 on page 1143, 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 135 on page 1143.

The procedure extends the example in Figure 132 on page 1137 in "Configuring request/reply to a cluster" on page 1137.

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

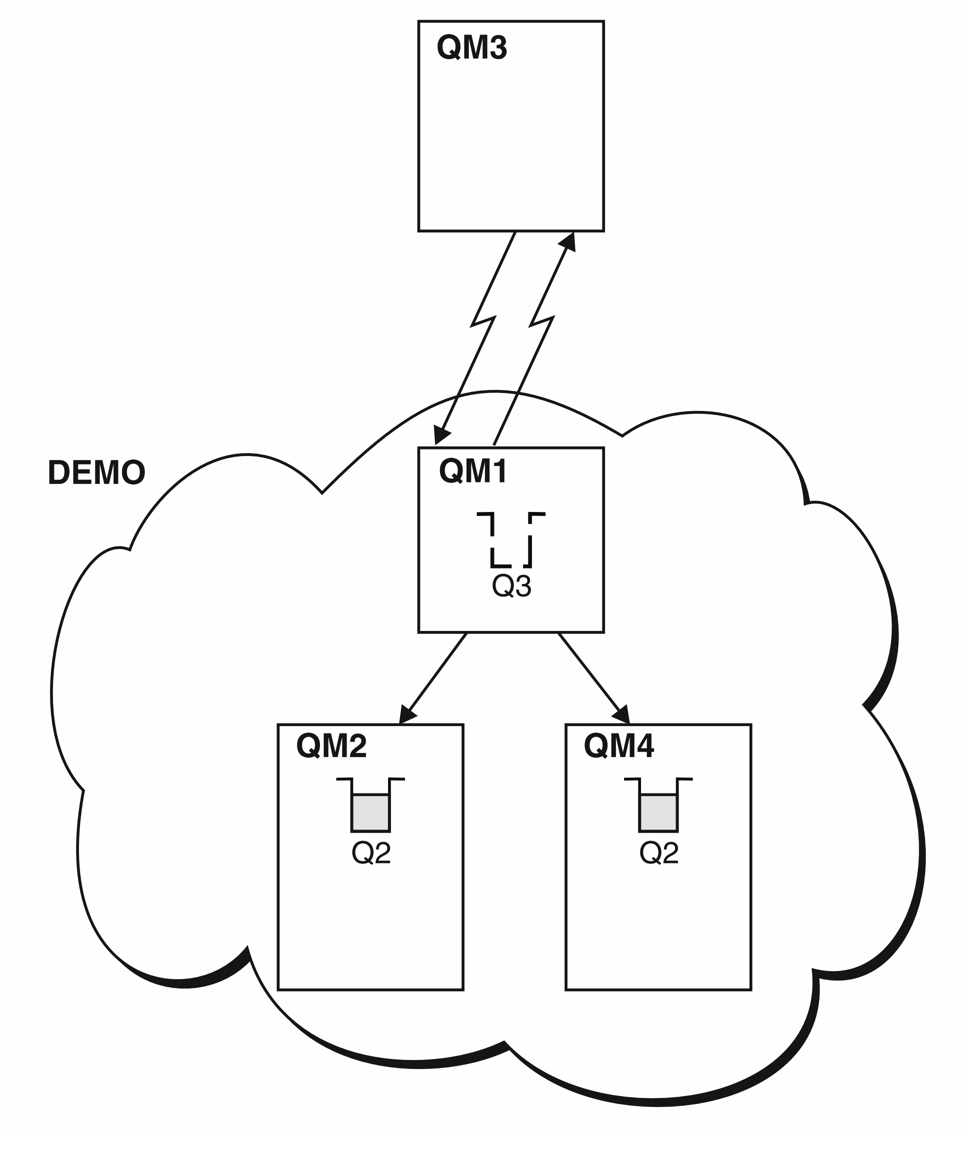


Figure 135. Putting from a queue manager outside the cluster

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 information:

Queue name resolution

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.

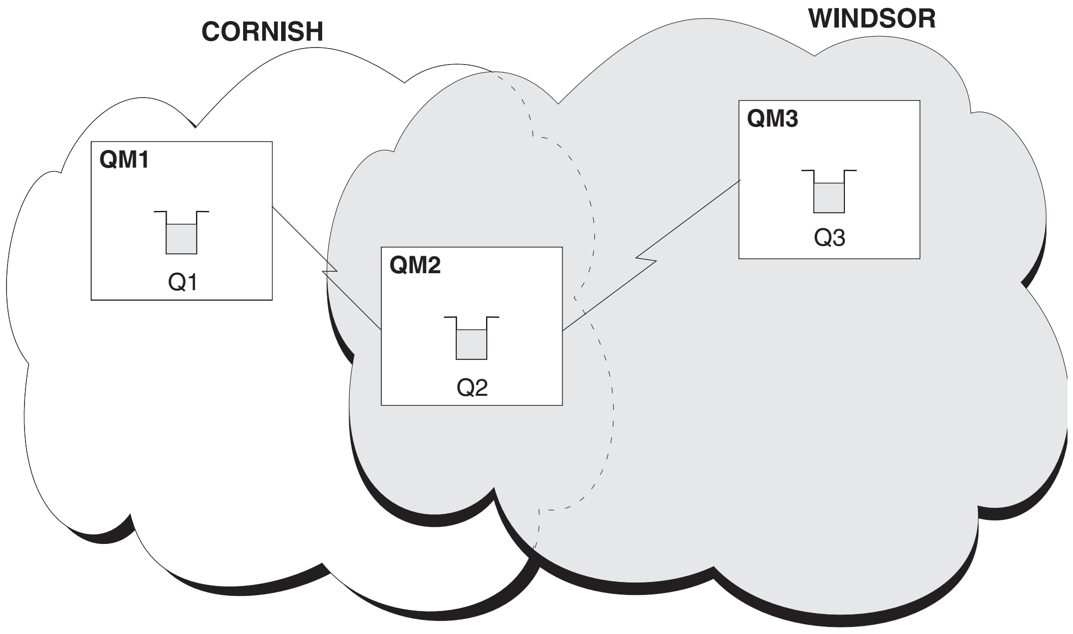


Figure 136. Bridging across clusters

Figure 136 shows two clusters with a bridge between them. There could be more than one bridge.

Configure the clusters using the following procedure:

Procedure

1. Define a cluster queue, Q1 on QM1.
   ```
   DEFINE QLOCAL(Q1) CLUSTER(CORNISH)
   ```
2. Define a cluster queue, Q3 on QM3.
   ```
   DEFINE QLOCAL(Q3) CLUSTER(WINDSOR)
   ```

3. Create a namelist called CORNISHWINDSOR on QM2, containing the names of both clusters.
   ```
   DEFINE NAMELIST(CORNISHWINDSOR) DESCR('CornishWindsor namelist')
   NAMES(CORNISH, WINDSOR)
   ```
4. Define a cluster queue, Q2 on QM2
   ```
   DEFINE QLOCAL(Q2) CLUSNL(CORNISHWINDSOR)
   ```

What to do next

QM2 is a member of both clusters and is the bridge between them. For each queue that you want to make visible across the bridge, you need a QALIAS definition on the bridge. For example in Figure 136 on page 1144, on QM2, you need:

```
DEFINE QALIAS(MYQ3) TARGQ(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.

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 1108, from which the reply message flow illustrated in figure 1, is taken.

The diagram shows the path taken by the reply message back to a temporary dynamic queue, which is called RQ. The server application, connected to QM3, opens the reply queue using the queue manager name QM2. The queue manager name QM2 is defined as a clustered queue manager alias on QM1. QM3 routes the reply message to QM1. QM1 routes the message to QM2.

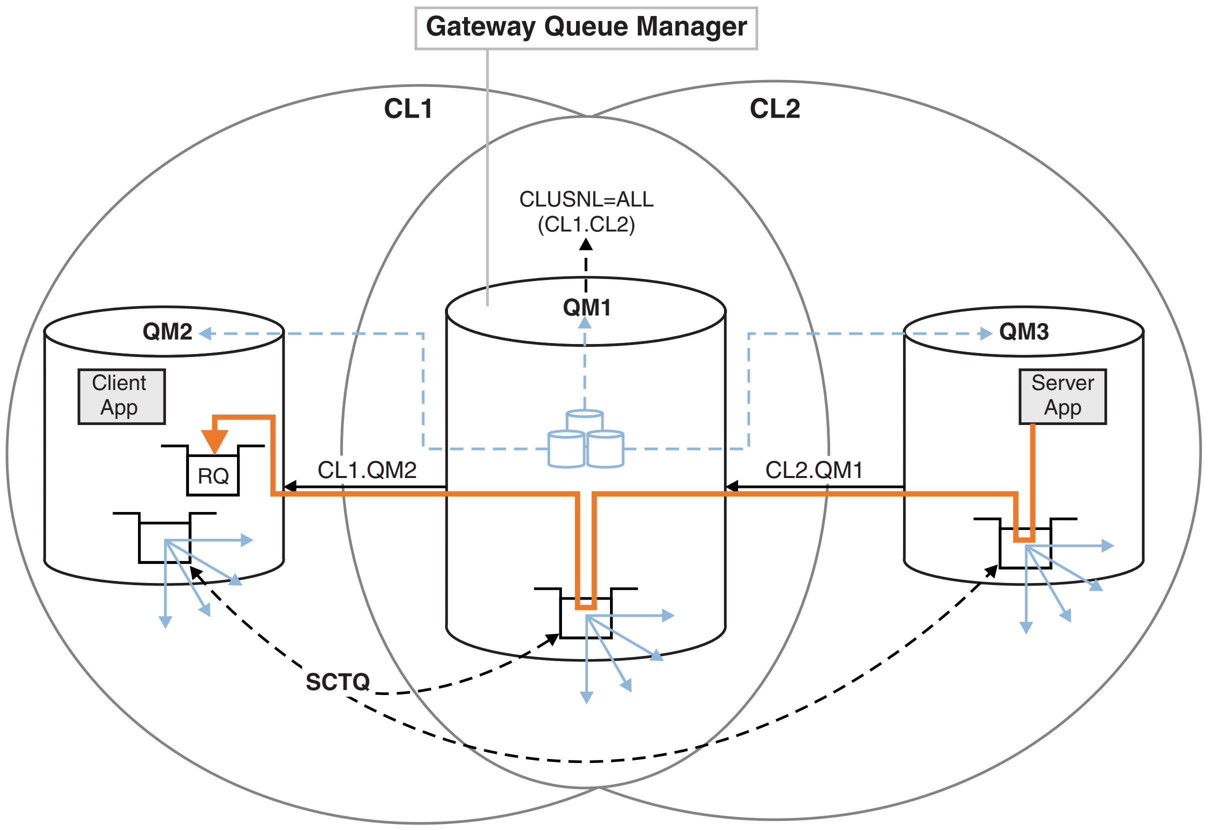


Figure 137. Using a queue manager alias to return the reply message to a different cluster

The way the routing works is as follows. Every queue manager in each cluster has a queue manager alias definition on QM1. The aliases are clustered in all the clusters. The grey dashed arrows from each of the aliases to a queue manager show that each queue manager alias is resolved to a real queue manager in at least one of the clusters. In this case, the QM2 alias is clustered in both cluster CL1 and CL2, and is resolved to the real queue manager QM2 in CL1. The server application creates the reply message using the reply to queue name RQ, and reply to queue manager name QM2. The message is routed to QM1 because the queue manager alias definition QM2 is defined on QM1 in cluster CL2 and queue manager QM2 is not in cluster CL2. As the message cannot be sent to the target queue manager, it is sent to the queue manager that has the alias definition.

QM1 places the message on the cluster transmission queue on QM1 for transferal to QM2. QM1 routes the message to QM2 because the queue manager alias definition on QM1 for QM2 defines QM2 as the real target queue manager. The definition is not circular, because alias definitions can refer only to real definitions; the alias cannot point to itself. The real definition is resolved by QM1, because both QM1 and QM2 are in the same cluster, CL1. QM1 finds out the connection information for QM2 from the repository for CL1, and routes the message to QM2. For the message to be rerouted by QM1, the server application must have opened the reply queue with the option DEFBIND set to MQBND_BIND_NOT_FIXED. If the server application had opened the reply queue with the option MQBND_BIND_ON_OPEN, the message is not rerouted and ends up on a dead letter queue.

Using a queue manager as a gateway into the cluster to workload balance messages from coming from outside the cluster.

You define a queue called EDINBURGH on more than one queue manager in the cluster. You want the clustering mechanism to balance the workload for messages coming to that queue from outside the cluster.

A queue manager from outside the cluster needs a transmit queue and sender-channel to one queue manager in the cluster. This queue is called a gateway queue manager. To take advantage of the default workload balancing mechanism, one of the following rules must apply:

- The gateway queue manager must not contain an instance of the EDINBURGH queue.
- The gateway queue manager specifies CLWLUSEQ(ANY) on ALTER QMGR.

For an example of workload balancing from outside a cluster, see “Configuring workload balancing from outside a cluster” on page 1142

Reply-to queue aliases and clusters:

A reply-to queue alias definition is used to specify alternative names for reply information. Reply-to queue alias definitions can be used with clusters just the same as in a distributed queuing environment.

For example:

- An application at queue manager VENICE sends a message to queue manager PISA using the MQPUT call. The application provides the following reply-to queue information in the message descriptor:
  ```
  ReplyToQ='QUEUE'
  ReplyToQMgr=''
  ```
- In order that replies sent to QUEUE can be received on OTHERQ at PISA, create a remote-queue definition on VENICE that is used as a reply-to queue alias. The alias is effective only on the system on which it was created.
  ```
  DEFINE QREMOTE(QUEUE) RNAME(OTHERQ) RQMNAME(PISA)
  ```

RQMNAME and QREMOTE can specify the same names, even if RQMNAME is itself a cluster queue manager.

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) TARGQ(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) TARGQ(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.

Using clusters for workload management

By defining multiple instances of a queue on different queue managers in a cluster you can spread the work of servicing the queue over multiple servers. There are several factors that can prevent messages being requeued to a different queue manager in the event of failure.

As well as setting up clusters to reduce system administration, you can create clusters in which more than one queue manager hosts an instance of the same queue.

You can organize your cluster such that the queue managers in it are clones of each other. Each queue manager is able to run the same applications and have local definitions of the same queues. z/OS For example, in a z/OS parallel sysplex the cloned applications might access data in a shared Db2 or Virtual Storage Access Method (VSAM) database. You can spread the workload between your queue managers by having several instances of an application. Each instance of the application receives messages and runs independently of the others.

The advantages of using clusters in this way are as follows:
- Increased availability of your queues and applications.
- Faster throughput of messages.
- More even distribution of workload in your network.

Any one of the queue managers that hosts an instance of a particular queue can handle messages destined for that queue, and applications do not name a queue manager when sending messages. If a cluster contains more than one instance of the same queue, IBM MQ selects a queue manager to route a message to. Suitable destinations are chosen based on the availability of the queue manager and queue, and on a number of cluster workload-specific attributes associated with queue managers, queues, and channels. See Workload balancing in clusters.

z/OS In IBM MQ for z/OS, queue managers that are in queue-sharing groups can host cluster queues as shared queues. Shared cluster queues are available to all queue managers in the same queue-sharing group. For example, in A cluster with multiple instances of the same queue, either or both of the queue managers `QM2` and `QM4` can be a shared-queue manager. Each has a definition for the queue `Q3`. Any of the queue managers in the same queue-sharing group as `QM4` can read a message put to the shared queue `Q3`. Each queue-sharing group can contain up to 32 queue managers, each with access to the same data. Queue-sharing significantly increases the throughput of your messages.

See the following subtopics for more information about cluster configurations for workload management:

Related concepts:

Comparison of clustering and distributed queuing

Distributed queuing and clusters

Components of a cluster

Cluster channels

"Routing messages to and from clusters" on page 1136
Use queue aliases, queue manager aliases, and remote queue definitions to connect clusters to external queue managers and other clusters.

Related tasks:

"Configuring a queue manager cluster" on page 1063
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 1074
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 1142
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:

What happens if a cluster queue is disabled for MQPUT

Workload balancing set on a cluster-sender channel is not working

Related information:

The Cluster Queue Monitoring sample program (AMQSCLM)

Writing and compiling cluster workload exits

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 138 on page 1151 shows a cluster in which there is more than one definition for the queue `Q3`. If an application at `QM1` puts a message to `Q3`, it does not necessarily know which instance of `Q3` is going to process its message. If an application is running on `QM2` or `QM4`, where there are local instances of `Q3`, the local instance of `Q3` is opened by default. By setting the `CLWLUSEQ` queue attribute, the local instance of the queue can be treated the same as a remote instance of the queue.

The MQOPEN option `DefBind` controls whether the target queue manager is chosen when the MQOPEN call is issued, or when the message is transferred from the transmission queue.

If you set `DefBind` to `MQBND_BIND_NOT_FIXED` the message can be sent to an instance of the queue that is available when the message is transmitted. This avoids the following problems:

- The target queue is unavailable when the message arrives at the target queue manager.
- The state of the queue has changed.
- The message has been put using a cluster queue alias, and no instance of the target queue exists on the queue manager where the instance of the cluster queue alias is defined.

If any if these problems are discovered at transmission time, another available instance of the target queue is sought and the message is rerouted. If no instances of the queue are available, the message is placed on the dead-letter queue.

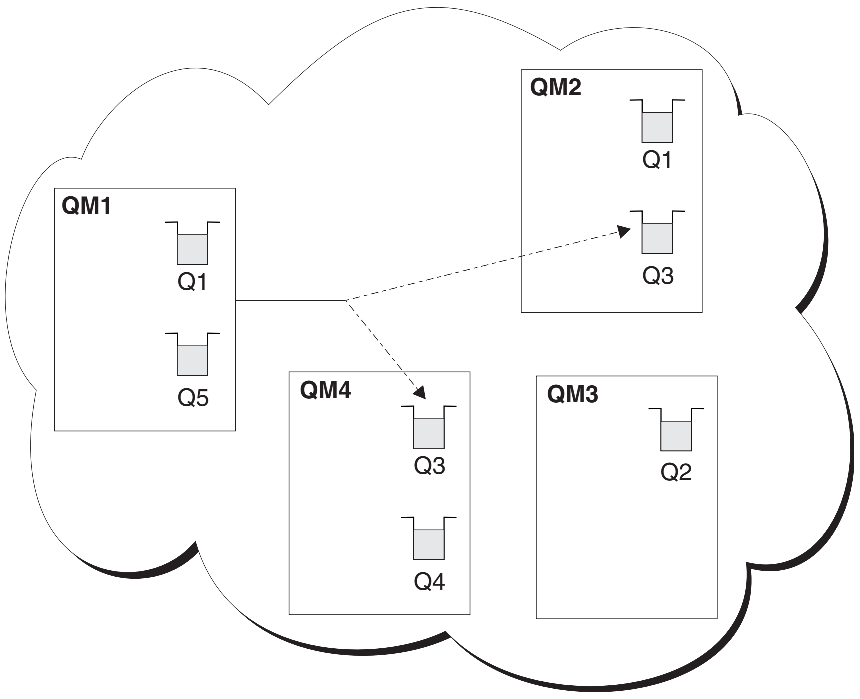


Figure 138. A cluster with multiple instances of the same queue

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.

Adding a queue manager that hosts a queue locally:

Follow these instructions to add an instance of INVENTQ to provide additional capacity to run the inventory application system in Paris and New York.

Before you begin

Note: For changes to a cluster to be propagated throughout the cluster, at least one full repository must always be available. Ensure that your repositories are available before starting this task.

Scenario:

- The INVENTORY cluster has been set up as described in Adding a new queue manager to a cluster. It contains three queue managers; LONDON and NEWYORK both hold full repositories, PARIS holds a partial repository. The inventory application runs on the system in New York, connected to the NEWYORK queue manager. The application is driven by the arrival of messages on the INVENTQ queue.
- We want to add an instance of INVENTQ to provide additional capacity to run the inventory application system in Paris and New York.

About this task

Follow these steps to add a queue manager that hosts a queue locally.

Procedure

1. Alter the PARIS queue manager.

   For the application in Paris to use the INVENTQ in Paris and the one in New York, we must inform the queue manager. On PARIS issue the following command:

   ```
   ALTER QMGR CLWLUSEQ(ANY)
   ```
2. Review the inventory application for message affinities.

   Before proceeding, ensure that the inventory application does not have any dependencies on the sequence of processing of messages. For more information, see Handling message affinities.
3. Install the inventory application on the system in Paris.
4. Define the cluster queue INVENTQ.

   The INVENTQ queue which is already hosted by the NEWYORK queue manager is also to be hosted by PARIS. Define it on the PARIS queue manager as follows:

   ```
   DEFINE QLOCAL(INVENTQ) CLUSTER(INVENTORY)
   ```

   Now that you have completed all the definitions, if you have not already done so, start the channel initiator on IBM MQ for z/OS. On all platforms, start a listener program on queue manager PARIS. The listener listens for incoming network requests and starts the cluster-receiver channel when it is needed.

Results

Figure 139 on page 1153 shows the cluster set up by this task.

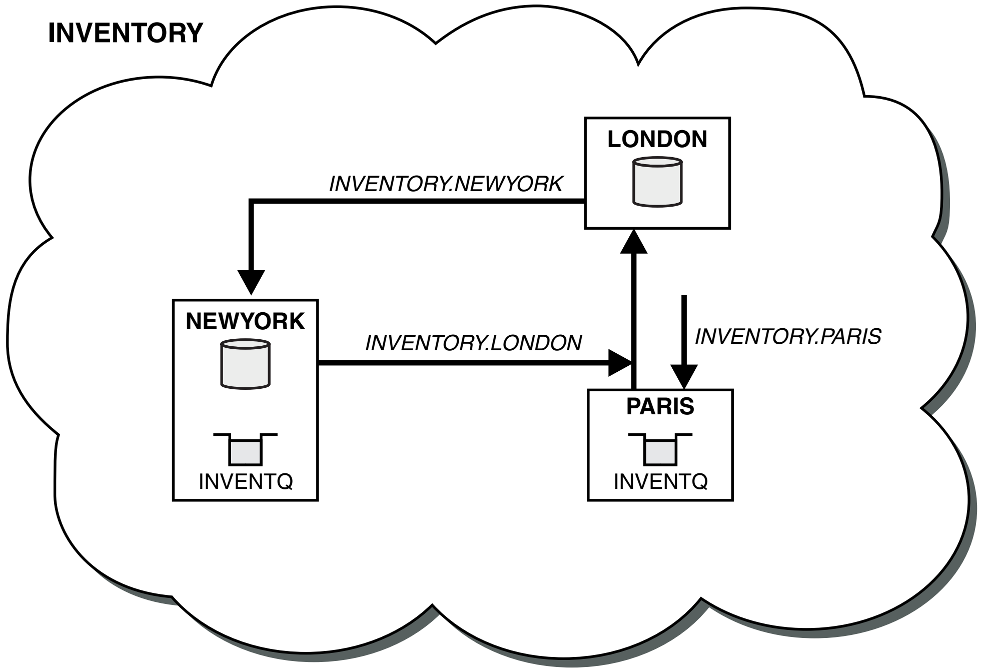


Figure 139. The `INVENTORY` cluster, with three queue managers

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.

Using two networks in a cluster:

Follow these instructions to add a new store in `TOKYO` where there are two different networks. Both need to be available for use to communicate with the queue manager in Tokyo.

Before you begin

Note: For changes to a cluster to be propagated throughout the cluster, at least one full repository must always be available. Ensure that your repositories are available before starting this task.

Scenario:

- The `INVENTORY` cluster has been set up as described in "Adding a queue manager to a cluster". It contains three queue managers; `LONDON` and `NEWYORK` both hold full repositories, `PARIS` holds a partial repository. The inventory application runs on the system in New York, connected to the `NEWYORK` queue manager. The application is driven by the arrival of messages on the `INVENTQ` queue.
- A new store is being added in `TOKYO` where there are two different networks. Both need to be available for use to communicate with the queue manager in Tokyo.

About this task

Follow these steps to use two networks in a cluster.

Procedure

1. Decide which full repository TOKYO refers to first.

   Every queue manager in a cluster must refer to one or other of the full repositories to gather information about the cluster. It builds up its own partial repository. It is of no particular significance which repository you choose. In this example, NEWYORK is chosen. Once the new queue manager has joined the cluster it communicates with both of the repositories.
2. Define the CLUSRCVR channels.

   Every queue manager in a cluster needs to define a cluster-receiver on which it can receive messages. This queue manager needs to be able to communicate on each network.

   ```
   DEFINE CHANNEL(INVENTORY.TOKYO.NETB) CHLTYPE(CLUSRCVR) TRPTYPE(TCP)
   CONNAME('TOKYO.NETB.CMSTORE.COM') CLUSTER(INVENTORY) DESCR('Cluster-receiver channel using
   network B for TOKYO')
   ```

   ```
   DEFINE CHANNEL(INVENTORY.TOKYO.NETA) CHLTYPE(CLUSRCVR) TRPTYPE(TCP)
   CONNAME('TOKYO.NETA.CMSTORE.COM') CLUSTER(INVENTORY) DESCR('Cluster-receiver channel using
   network A for TOKYO')
   ```
3. Define a CLUSSDR channel on queue manager TOKYO.

   Every queue manager in a cluster needs to define one cluster-sender channel on which it can send messages to its first full repository. In this case we have chosen NEWYORK, so TOKYO needs the following definition:

   ```
   DEFINE CHANNEL(INVENTORY.NEWYORK) CHLTYPE(CLUSSDR) TRPTYPE(TCP) CONNAME(NEWYORK.CHSTORE.COM)
   CLUSTER(INVENTORY) DESCR('Cluster-sender channel from TOKYO to repository at NEWYORK')
   ```

   Now that you have completed all the definitions, if you have not already done so start the channel initiator on IBM MQ for z/OS. On all platforms, start a listener program on queue manager PARIS. The listener program listens for incoming network requests and starts the cluster-receiver channel when it is needed.

Results

Figure 140 on page 1155 shows the cluster set up by this task.

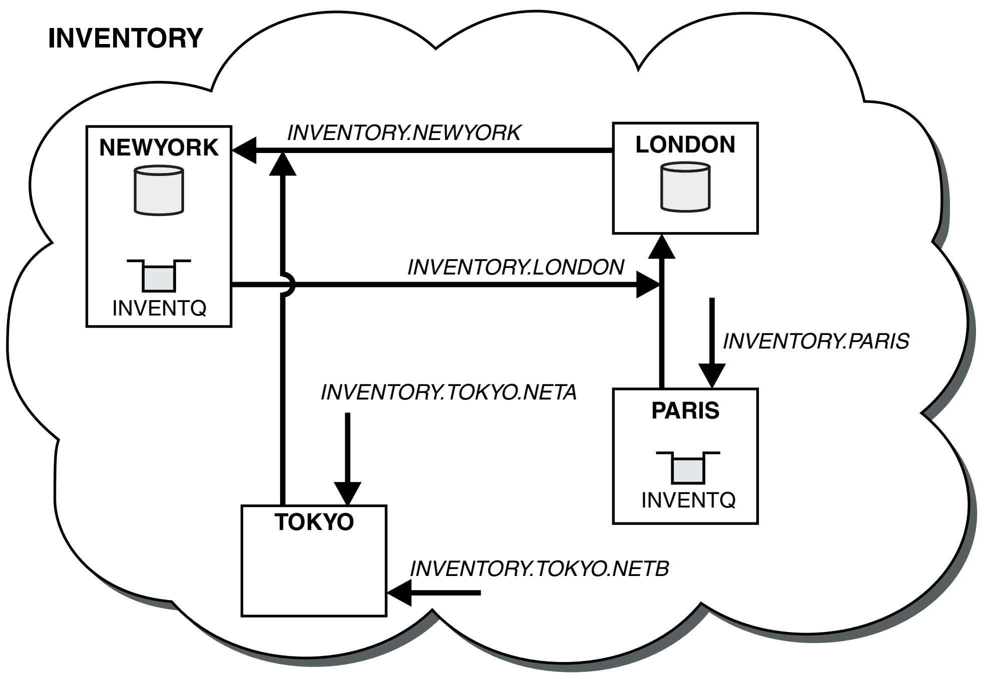


Figure 140. The `INVENTORY` cluster, with four queue managers

By making only three definitions, we have added the queue manager `TOKYO` to the cluster with two different network routes available to it.

Related tasks:

"Adding a queue manager to a cluster" on page 1085
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 1153. 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:

1. `ALTER CHANNEL(INVENTORY.TOKYO.NETA) CHLTYPE(CLUSRCVR) NETPRTY(2) DESCR('Main cluster-receiver channel for TOKYO')`
2. `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.

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 1085. 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 141 shows the cluster set up by this task.

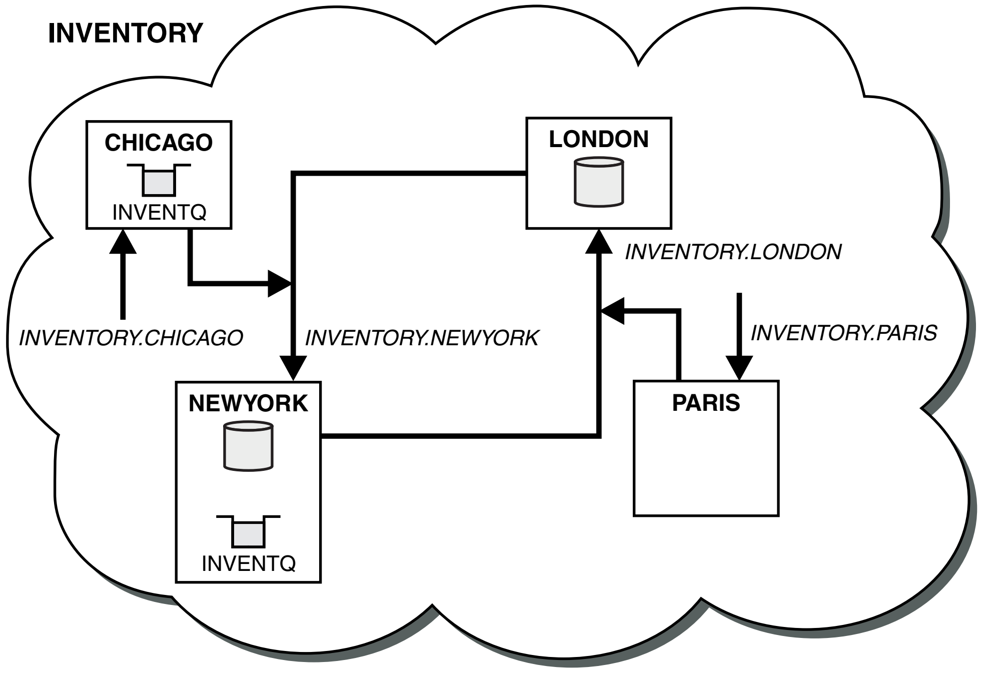


Figure 141. The INVENTORY cluster, with four queue managers

The INVENTQ queue and the inventory application are now hosted on two queue managers in the cluster. The CHICAGO queue manager is a backup. Messages put to INVENTQ are routed to NEWYORK unless it is unavailable when they are sent instead to CHICAGO.

Note:

The availability of a remote queue manager is based on the status of the channel to that queue manager. When channels start, their state changes several times, with some of the states being less preferential to the cluster workload management algorithm. In practice this means that lower priority (backup) destinations can be chosen while the channels to higher priority (primary) destinations are starting.

If you need to ensure that no messages go to a backup destination, do not use CLWLPRTY. Consider using separate queues, or CLWLRANK with a manual switch over from the primary to back up.

Restricting the number of channels used:

Follow these instructions to restrict the number of active channels each server runs when a price check application is installed on various queue managers.

Before you begin

Note: For changes to a cluster to be propagated throughout the cluster, at least one full repository must always be available. Ensure that your repositories are available before starting this task.

Scenario:

- A price check application is to be installed on various queue managers. To keep the number of channels being used to a low number, the number of active channels each server runs is restricted. The application is driven by the arrival of messages on the PRICEQ queue.
- Four server queue managers host the price check application. Two query queue managers send messages to the PRICEQ to query a price. Two more queue managers are configured as full repositories.

About this task

Follow these steps to restrict the number of channels used.

Procedure

1. Choose two full repositories.

   Choose two queue managers to be the full repositories for your price check cluster. They are called REPOS1 and REPOS2.

   Issue the following command:

   ```
   ALTER QMGR REPOS(PRICECHECK)
   ```
2. Define a CLUSRCVR channel on each queue manager.

   At each queue manager in the cluster, define a cluster-receiver channel and a cluster-sender channel. It does not matter which is defined first.

   ```
   DEFINE CHANNEL(PRICECHECK.SERVE1) CHLTYPE(CLUSRCVR) TRPTYPE(TCP) CONNAME(SERVER1.COM)
   CLUSTER(PRICECHECK) DESCR('Cluster-receiver channel')
   ```
3. Define a CLUSSDR channel on each queue manager.

   Make a CLUSSDR definition at each queue manager to link that queue manager to one or other of the full repository queue managers.

   ```
   DEFINE CHANNEL(PRICECHECK.REPOS1) CHLTYPE(CLUSSDR) TRPTYPE(TCP) CONNAME(REPOS1.COM)
   CLUSTER(PRICECHECK) DESCR('Cluster-sender channel to repository queue manager')
   ```
4. Install the price check application.
5. Define the PRICEQ queue on all the server queue managers.

Issue the following command on each:

```
DEFINE QLOCAL(PRICEQ) CLUSTER(PRICECHECK)
```

6. Restrict the number of channels used by queries

   On the query queue managers we restrict the number of active channels used, by issuing the following commands on each:

   ```
   ALTER QMGR CLWLMRUC(2)
   ```

7. If you have not already done so, start the channel initiator on IBM MQ for z/OS. On all platforms, start a listener program.

   The listener program listens for incoming network requests and starts the cluster-receiver channel when it is needed.

Results

Figure 142 shows the cluster set up by this task.

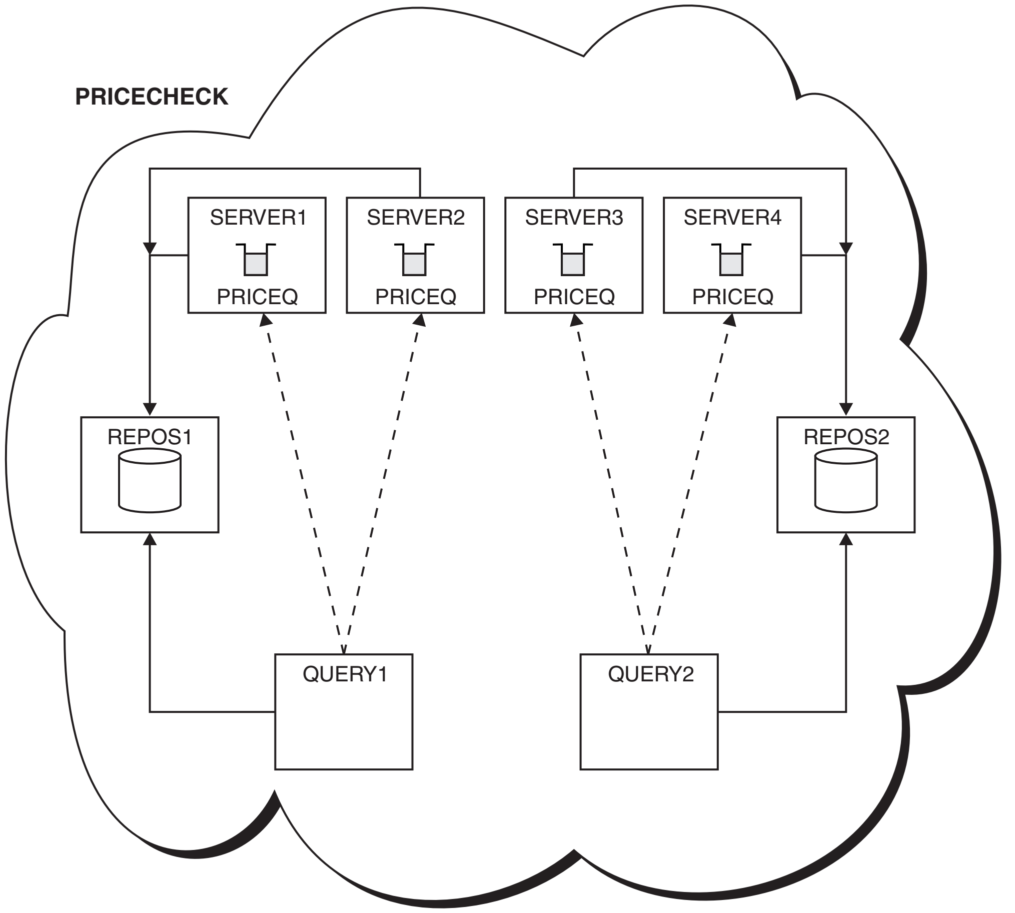


Figure 142. The `PRICECHECK` *cluster, with four server queue managers, two repositories, and two query queue managers*

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`.

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 1085. 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 1160 shows the cluster set up by this task.

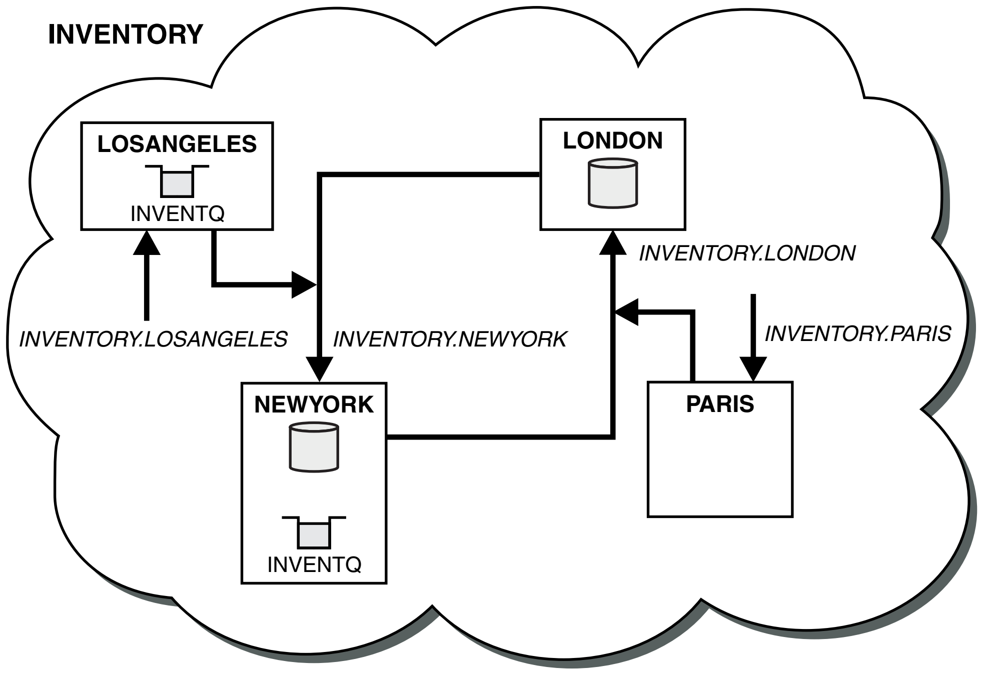


Figure 143. The INVENTORY cluster with four queue managers

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.

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)`.

Handling message affinities:

Message affinities are rarely part of good programming design. You need to remove message affinities to use clustering fully. If you cannot remove message affinities, you can force related messages to be delivered using the same channel and to the same queue manager.

If you have applications with message affinities, remove the affinities before starting to use clusters.

Removing message affinities improves the availability of applications. An application sends a batch of messages that has message affinities to a queue manager. The queue manager fails after receiving only part of the batch. The sending queue manager must wait for it to recover and process the incomplete message batch before it can send any more messages.

Removing messages affinities also improves the scalability of applications. A batch of messages with affinities can lock resources at the destination queue manager while waiting for subsequent messages. These resources might remain locked for long periods of time, preventing other applications from doing their work.

Furthermore, message affinities prevent the cluster workload management routines from making the best choice of queue manager.

To remove affinities, consider the following possibilities:

- Carrying state information in the messages
- Maintaining state information in nonvolatile storage accessible to any queue manager, for example in a Db2 database
- Replicating read-only data so that it is accessible to more than one queue manager

If it is not appropriate to modify your applications to remove message affinities, there are a number of possible solutions to the problem.

Name a specific destination on the MQOPEN call

Specify the remote-queue name and the queue manager name on each MQOPEN call, and all messages put to the queue using that object handle go to the same queue manager, which might be the local queue manager.

Specifying the remote-queue name and the queue manager name on each MQOPEN call has disadvantages:

- No workload balancing is carried out. You do not take advantage of the benefits of cluster workload balancing.
- If the target queue manager is remote and there is more than one channel to it, the messages might take different routes and the sequence of messages is still not preserved.
- If your queue manager has a definition for a transmission queue with the same name as the destination queue manager, messages go on that transmission queue rather than on the cluster transmission queue.

Return the queue manager name in the reply-to queue manager field

Allow the queue manager that receives the first message in a batch to return its name in its response. It does this using the `ReplyToQMgr` field of the message descriptor. The queue manager at the sending end can then extract the reply-to queue manager name and specify it on all subsequent messages.

Using the `ReplyToQMgr` information from the response has disadvantages:

- The requesting queue manager must wait for a response to its first message
- You must write additional code to find and use the `ReplyToQMgr` information before sending subsequent messages
- If there is more than one route to the queue manager, the sequence of the messages might not be preserved

Set the `MQOO_BIND_ON_OPEN` option on the MQOPEN call

Force all your messages to be put to the same destination using the `MQOO_BIND_ON_OPEN` option on the MQOPEN call. Either `MQOO_BIND_ON_OPEN` or `MQOO_BIND_ON_GROUP` must be specified when using message groups with clusters to ensure that all messages in the group are processed at the same destination.

By opening a queue and specifying `MQOO_BIND_ON_OPEN`, you force all messages that are sent to this queue to be sent to the same instance of the queue. `MQOO_BIND_ON_OPEN` binds all messages to the same queue manager and also to the same route. For example, if there is an IP route and a NetBIOS route to the same destination, one of these is selected when the queue is opened and this selection is honored for all messages put to the same queue using the object handle obtained.

By specifying `MQOO_BIND_ON_OPEN` you force all messages to be routed to the same destination. Therefore applications with message affinities are not disrupted. If the destination is not available, the messages remain on the transmission queue until it becomes available again.

MQOO_BIND_ON_OPEN also applies when the queue manager name is specified in the object descriptor when you open a queue. There might be more than one route to the named queue manager. For example, there might be multiple network paths or another queue manager might have defined an alias. If you specify MQOO_BIND_ON_OPEN, a route is selected when the queue is opened.

Note: This is the recommended technique. However, it does not work in a multi-hop configuration in which a queue manager advertises an alias for a cluster queue. Nor does it help in situations in which applications use different queues on the same queue manager for different groups of messages.

An alternative to specifying MQOO_BIND_ON_OPEN on the MQOPEN call, is to modify your queue definitions. On your queue definitions, specify DEFBIND(OPEN), and allow the DefBind option on the MQOPEN call to default to MQOO_BIND_AS_Q_DEF.

Set the MQOO_BIND_ON_GROUP option on the MQOPEN call

Force all your messages in a group to be put to the same destination using the MQOO_BIND_ON_GROUP option on the MQOPEN call. Either MQOO_BIND_ON_OPEN or MQOO_BIND_ON_GROUP must be specified when using message groups with clusters to ensure that all messages in the group are processed at the same destination.

By opening a queue and specifying MQOO_BIND_ON_GROUP, you force all messages in a group that are sent to this queue to be sent to the same instance of the queue. MQOO_BIND_ON_GROUP binds all messages in a group to the same queue manager, and also to the same route. For example, if there is an IP route and a NetBIOS route to the same destination, one of these is selected when the queue is opened and this selection is honored for all messages in a group put to the same queue using the object handle obtained.

By specifying MQOO_BIND_ON_GROUP you force all messages in a group to be routed to the same destination. Therefore applications with message affinities are not disrupted. If the destination is not available, the messages remain on the transmission queue until it becomes available again.

MQOO_BIND_ON_GROUP also applies when the queue manager name is specified in the object descriptor when you open a queue. There might be more than one route to the named queue manager. For example, there might be multiple network paths or another queue manager might have defined an alias. If you specify MQOO_BIND_ON_GROUP, a route is selected when the queue is opened.

If MQOO_BIND_ON_GROUP is specified but the messages are not grouped, the behavior is equivalent to MQOO_BIND_NOT_FIXED.

Note: This is the recommended technique for ensuring that messages in a group are sent to the same destination. However, it does not work in a multi-hop configuration in which a queue manager advertises an alias for a cluster queue.

An alternative to specifying MQOO_BIND_ON_GROUP on the MQOPEN call, is to modify your queue definitions. On your queue definitions, specify DEFBIND(GROUP), and allow the DefBind option on the MQOPEN call to default to MQOO_BIND_AS_Q_DEF.

Write a customized cluster workload exit program

Instead of modifying your applications you can circumvent the message affinities problem by writing a cluster workload exit program. Writing a cluster workload exit program is not easy and is not a recommended solution. The program would have to be designed to recognize the affinity by inspecting the content of messages. Having recognized the affinity, the program would have to force the workload management utility to route all related messages to the same queue manager.

Configuring publish/subscribe messaging

You can start, stop and display the status of queued publish/subscribe. You can also add and remove streams, and add and delete queue managers from a broker hierarchy.

Procedure

See the following subtopics for more information on controlling queued publish/subscribe:

- "Setting queued publish/subscribe message attributes"
- "Starting queued publish/subscribe" on page 1166
- "Stopping queued publish/subscribe" on page 1167
- "Adding a stream" on page 1167
- "Deleting a stream" on page 1168
- "Adding a subscription point" on page 1169
- "Combining topic spaces in publish/subscribe networks" on page 1175

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 123. Publish/subscribe configuration parameters

Description	MQSC parameter name
Command message retry count	**PSRTYCNT**
Discard undeliverable command input message	**PSNPMSG**
Behavior following undeliverable command response message	**PSNPRES**
Process command messages under syncpoint	**PSSYNCPT**

The Broker stanza is used to manage the following configuration settings:

- `PersistentPublishRetry=yes | force`

  If you specify `Yes`, then if a publication of a persistent message through the queued publish/subscribe interface fails, and no negative reply was requested, the publish operation is retried.

  If you requested a negative response message, the negative response is sent and no further retry occurs.

  If you specify `Force`, then if a publication of a persistent message through the queued publish/subscribe interface fails, the publish operation is retried until the it is successfully processed. No negative response is sent.

- `NonPersistentPublishRetry=yes | force`

  If you specify `Yes`, then if a publication of a non-persistent message through the queued publish/subscribe interface fails, and no negative reply was requested, the publish operation is retried.

  If you requested a negative response message, the negative response is sent and no further retry occurs.

  If you specified `Force`, then if a publication of a non-persistent message through the queued publish/subscribe interface fails, the publish operation is retried until it is successfully processed. No negative response is sent.

Note: If you want to enable this functionality for non-persistent messages, then as well as setting the `NonPersistentPublishRetry` value you must also ensure that the queue manager attribute **`PSSYNCPT`** is set to `Yes`.

Doing this might also have an impact on the performance of processing non-persistent publications as the **`MQGET`** from the STREAM queue now occurs under syncpoint.

- `PublishBatchSize=`*number*

  The broker normally processes publish messages within syncpoint. It can be inefficient to commit each publication individually, and in some circumstances the broker can process multiple publish messages in a single unit of work. This parameter specifies the maximum number of publish messages that can be processed in a single unit of work

  The default value for `PublishBatchSize` is 5.
- `PublishBatchInterval=`*number*

  The broker normally processes publish messages within syncpoint. It can be inefficient to commit each publication individually, and in some circumstances the broker can process multiple publish messages in a single unit of work. This parameter specifies the maximum time (in milliseconds) between the first message in a batch and any subsequent publication included in the same batch.

  A batch interval of 0 indicates that up to `PublishBatchSize` messages can be processed, provided that the messages are available immediately.

  The default value for `PublishBatchInterval` is zero.

Procedure

Use IBM MQ Explorer, programmable commands, or the **`runmqsc`** command to alter the queue manager attributes that control the behavior of publish/subscribe.

Example

ALTER QMGR PSNPRES(SAFE)

Starting queued publish/subscribe

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 Version 6 publish/subscribe hierarchies.

Before you begin

Familiarize yourself with the way publish/subscribe streams operate. See Streams and topics.

About this task

Use PCF command, **`runmqsc`**, or 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 Version 6 stream.
2. Define a local topic with the same name as the Version 6 stream.
3. Add the name of the queue to the namelist, `SYSTEM.QPUBSUB.QUEUE.NAMELIST`

4. Repeat for all queue managers at Version 7.1 or above that are in the publish/subscribe hierarchy.

Adding `'Sport'`

In the example of sharing the stream `'Sport'`, Version 6 and Version 7.1 queue managers are working in the same publish/subscribe hierarchy. The Version 6 queue managers share a stream called `'Sport'`. The example shows how to create a queue and a topic on Version 7.1 queue managers called `'Sport'`, with a topic string `'Sport'` that is shared with the Version 6 stream `'Sport'`.

A Version 7.1 publish application, publishing to topic `'Sport'`, with topic string `'Soccer/Results'`, creates the resultant topic string `'Sport/Soccer/Results'`. On Version 7.1 queue managers, subscribers to topic `'Sport'`, with topic string `'Soccer/Results'` receive the publication.

On Version 6 queue managers, subscribers to stream `'Sport'`, with topic string `'Soccer/Results'` receive the publication.

```
runmqsc QM1
5724-H72 (C) Copyright IBM Corp. 1994, 2008.  ALL RIGHTS RESERVED.
Starting MQSC for queue manager QM1.
define qlocal('Sport')
     1 : define qlocal('Sport')
AMQ8006: IBM MQ queue created.
define topic('Sport') topicstr('Sport')
     2 : define topic('Sport') topicstr('Sport')
AMQ8690: IBM MQ topic created.
alter namelist(SYSTEM.QPUBSUB.QUEUE.NAMELIST) NAMES('Sport', 'SYSTEM.BROKER.DEFAULT.STREAM', 'SYSTEM.BROKER.ADMIN.STREAM')
     3 : alter namelist(SYSTEM.QPUBSUB.QUEUE.NAMELIST) NAMES('Sport', 'SYSTEM.BROKER.DEFAULT.STREAM', 'SYSTEM.BROKER.ADMIN.STREAM')
AMQ8551: IBM MQ namelist changed.
```

Note: You need both to provide the existing names in the namelist object, as well as the new names that you are adding, to the **`alter namelist`** command.

What to do next

Information about the stream is passed to other brokers in the hierarchy.

If a broker is Version 6, administer it as a Version 6 broker. That is, you have a choice of creating the stream queue manually, or letting the broker create the stream queue dynamically when it is needed. The queue is based on the model queue definition, `SYSTEM.BROKER.MODEL.STREAM`.

If a broker is Version 7.1, you must configure each Version 7.1 queue manager in the hierarchy manually.

Deleting a stream

You can delete a stream from an IBM MQ Version 7.1, or later, queue manager.

Before you begin

Before deleting a stream you must ensure that there are no remaining subscriptions to the stream and quiesce all applications that use the stream. If publications continue to flow to a deleted stream, it takes a lot of administrative effort to restore the system to a cleanly working state.

Procedure

1. Find all the connected brokers that host this stream.
2. Cancel all subscriptions to the stream on all the brokers.
3. Remove the queue (with the same name as the stream) from the namelist, `SYSTEM.QPUBSUB.QUEUE.NAMELIST`.
4. Delete or purge all the messages from the queue with the same name as the stream.

5. Delete the queue with the same name as the stream.
6. Delete the associated topic object.

What to do next

Repeat steps 3 to 5 on all the other connected Version 7.1, or later, queue managers hosting the stream.

Adding a subscription point

How to extend an existing queued publish/subscribe application that you have migrated from an earlier version of IBM Integration Bus with a new subscription point.

Before you begin

1. Check that the subscription point is not already defined in `SYSTEM.QPUBSUB.SUBPOINT.NAMELIST`.
2. Check if there is a topic object or a topic string with the same name as the subscription point.

About this task

IBM WebSphere MQ Version 7.1, or later, applications do not use subscription points, but they can interoperate with existing applications that do, using the subscription point migration mechanism.

Important: The subscription point migration mechanism has been removed from IBM MQ Version 8.0. If you need to migrate your existing applications, you must carry out the procedures described in the documentation for your version of the product, before you migrate to the latest version.

Subscription points do not work with queued publish/subscribe programs that use `MQRFH1` headers, which have been migrated from IBM MQ Version 6, or earlier.

There is no need to add subscription points to use integrated publish/subscribe applications written for IBM MQ Version 7.1, or later.

Procedure

1. Add the name of the subscription point to `SYSTEM.QPUBSUB.SUBPOINT.NAMELIST`.
   - On z/OS, the **`NLTYPE`** is `NONE`, the default.
   - Repeat the step on every queue manager that is connected in the same publish/subscribe topology.
2. Add a topic object, preferably giving it the name of the subscription point, with a topic string matching the name of the subscription point.
   - If the subscription point is in a cluster, add the topic object as a cluster topic on the cluster topic host.
   - If a topic object exists with the same topic string as the name of the subscription point, use the existing topic object. You must understand the consequences of the subscription point reusing an existing topic. If the existing topic is part of an existing application, you must resolve the collision between two identically named topics.
   - If a topic object exists with the same name as the subscription point, but a different topic string, create a topic with a different name.
3. Set the **`Topic`** attribute `WILDCARD` to the value `BLOCK`.

   Blocking subscriptions to # or * isolates wildcard subscriptions to subscription points, see Wildcards and subscription points.
4. Set any attributes that you require in the topic object.

Example

The example shows a **`runmqsc`** command file that adds two subscription points, `USD` and `GBP`.

```
DEFINE TOPIC(USD) TOPICSTR(USD)
DEFINE TOPIC(GBP) TOPICSTR(GBP) WILDCARD(BLOCK)
ALTER NL(SYSTEM.QPUBSUB.SUBPOINT.NAMELIST) NAMES(SYSTEM.BROKER.DEFAULT.SUBPOINT, USD, GBP)
```

Note:

1. Include the default subscription point in the list of subscription points added using the **ALTER** command. **ALTER** deletes existing names in the namelist.
2. Define the topics before altering the namelist. The queue manager only checks the namelist when the queue manager starts and when the namelist is altered.

Configuring distributed publish/subscribe networks

Queue managers that are connected together into a distributed publish/subscribe topology share a common federated topic space. Subscriptions created on one queue manager can receive messages published by an application connected to another queue manager in the topology.

You can control the extent of topic spaces created by connecting queue managers together in clusters or hierarchies. In a publish/subscribe cluster, a topic object must be 'clustered' for each branch of the topic space that is to span the cluster. In a hierarchy, each queue manager must be configured to identify its 'parent' in the hierarchy.

You can further control the flow of publications and subscriptions within the topology by choosing whether each publication and subscription is either local or global. Local publications and subscriptions are not propagated beyond the queue manager to which the publisher or subscriber is connected.

Related information:

Distributed publish/subscribe networks

Publication scope

Subscription scope

Topic spaces

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 1085. It contains three queue managers; LONDON and NEWYORK both hold full repositories, PARIS holds a partial repository.

About this task

When you define a topic on a queue manager in a cluster, you need to specify whether the topic is a cluster topic, and (if so) the routing within the cluster for publications and subscriptions for this topic. To make the topic a cluster topic, you configure the **CLUSTER** property on the TOPIC object with the name of the cluster. By defining a cluster topic on a queue manager in the cluster, you make the topic available to the whole cluster. To choose the message routing to use within the cluster, you set the **CLROUTE** property on the TOPIC object to one of the following values:

- **DIRECT**
- **TOPICHOST**

By default, topic routing is **DIRECT**. This was the only option prior to IBM MQ Version 8.0. When you configure a direct routed clustered topic on a queue manager, all queue managers in the cluster become aware of all other queue managers in the cluster. When performing publish and subscribe operations, each queue manager can connect direct to any other queue manager in the cluster. See Direct routed publish/subscribe clusters.

From IBM MQ Version 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 1173.

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 1177. You can also control which publications flow from one cluster to another. See "Combining and isolating topic spaces in multiple clusters" on page 1179.

Related information:

Designing publish/subscribe clusters

Distributed publish/subscribe troubleshooting

Inhibiting clustered publish/subscribe

Moving a cluster topic definition to a different queue manager

For either topic host routed or direct routed clusters, you might need to move a cluster topic definition when decommissioning a queue manager, or because a cluster queue manager has failed or is unavailable for a significant period of time.

About this task

You can have multiple definitions of the same cluster topic object in a cluster. This is a normal state for a topic host routed cluster, and an unusual state for a direct routed cluster. For more information, see Multiple cluster topic definitions of the same name.

To move a cluster topic definition to a different queue manager in the cluster without interrupting the flow of publications, complete the following steps. The procedure moves a definition from queue manager `QM1` to queue manager `QM2`.

Procedure

1. Create a duplicate of the cluster topic definition on `QM2`.

   For direct routing, set all the attributes to match the definition of `QM1`.

   For topic host routing, initially define the new topic host as `PUB(DISABLED)`. This allows `QM2` to learn of the subscriptions in the cluster, but not to start routing publications.
2. Wait for information to be propagated through the cluster.

   Wait for the new cluster topic definition to be propagated by the full repository queue managers to all queue managers in the cluster. Use the **DISPLAY CLUSTER** command to display the cluster topics on each cluster member, and check for a definition originating from `QM2`.

   For topic host routing, wait for the new topic host on `QM2` to learn of all subscriptions. Compare the proxy subscriptions known to `QM2` and those known to `QM1`. One way to view the proxy subscriptions on a queue manager is to issue the following **runmqsc** command:

   ```
   DISPLAY SUB(*) SUBTYPE(PROXY)
   ```
3. For topic host routing, redefine the topic host on `QM2` as `PUB(ENABLED)`, then redefine the topic host on `QM1` as `PUB(DISABLED)`.

   Now that the new topic host on `QM2` has learnt 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.

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 1170.

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.

Combining publication and subscription scopes

From IBM WebSphere MQ Version 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 information:

Distributed publish/subscribe networks

Publication scope

Subscription scope

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 information:

Distributed publish/subscribe networks

Topic spaces

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 information:

Distributed publish/subscribe networks

Topic spaces

Defining cluster topics

Adding a version 7 or later queue manager to existing Version 6 topic spaces:

Extend an existing Version 6 publish/subscribe system to interoperate with a version 7 or later queue manager, sharing the same topic spaces.

Before you begin

You have an existing Version 6 publish/subscribe system.

You have installed IBM MQ version 7 or later on a new server and configured a queue manager.

About this task

You want to extend your existing Version 6 publish/subscribe system to work with version 7 or later queue managers.

You have decided to stabilize development of the Version 6 publish/subscribe system that uses the queued publish/subscribe interface. You intend to add extensions to the system using the version 7 or later MQI. You have no plans now to rewrite the queued publish/subscribe applications.

You intend to upgrade the Version 6 queue managers to version 7 or later in the future. For now, you are continuing to run the existing queued publish/subscribe applications on the version 7 or later queue managers.

Procedure

1. Create one set of sender-receiver channels to connect the version 7 or later queue manager with one of the Version 6 queue managers in both directions.
2. Create two transmission queues with the names of the target queue managers. Use queue manager aliases if you cannot use the name of the target queue manager as the transmission queue name for some reason.
3. Configure the transmission queues to trigger the sender channels.
4. If the Version 6 publish/subscribe system uses streams, add the streams to the version 7 or later queue manager as described in Adding a stream.
5. Check the version 7 or later queue manager **PSMODE** is set to ENABLE.

6. Alter its **PARENT** attribute to refer to one of the Version 6 queue managers.
7. Check the status of the parent-child relationship between the queue managers is active in both directions.

What to do next

Once you have completed the task, both the Version 6 and version 7 or later queue manager share the same topic spaces. For example, you can do all the following tasks.
- Exchange publications and subscriptions between Version 6 and version 7 or later queue managers.
- Run your existing Version 6 publish/subscribe programs on the version 7 or later queue manager.
- View and modify the topic space on either the Version 6 or version 7 or later queue manager.
- Write version 7 or later publish/subscribe applications and run them on the version 7 or later queue manager.
- Create new publications and subscriptions with the version 7 or later applications and exchange them with Version 6 applications.

Related information:

Distributed publish/subscribe networks

Topic spaces

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 144 on page 1178. 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.

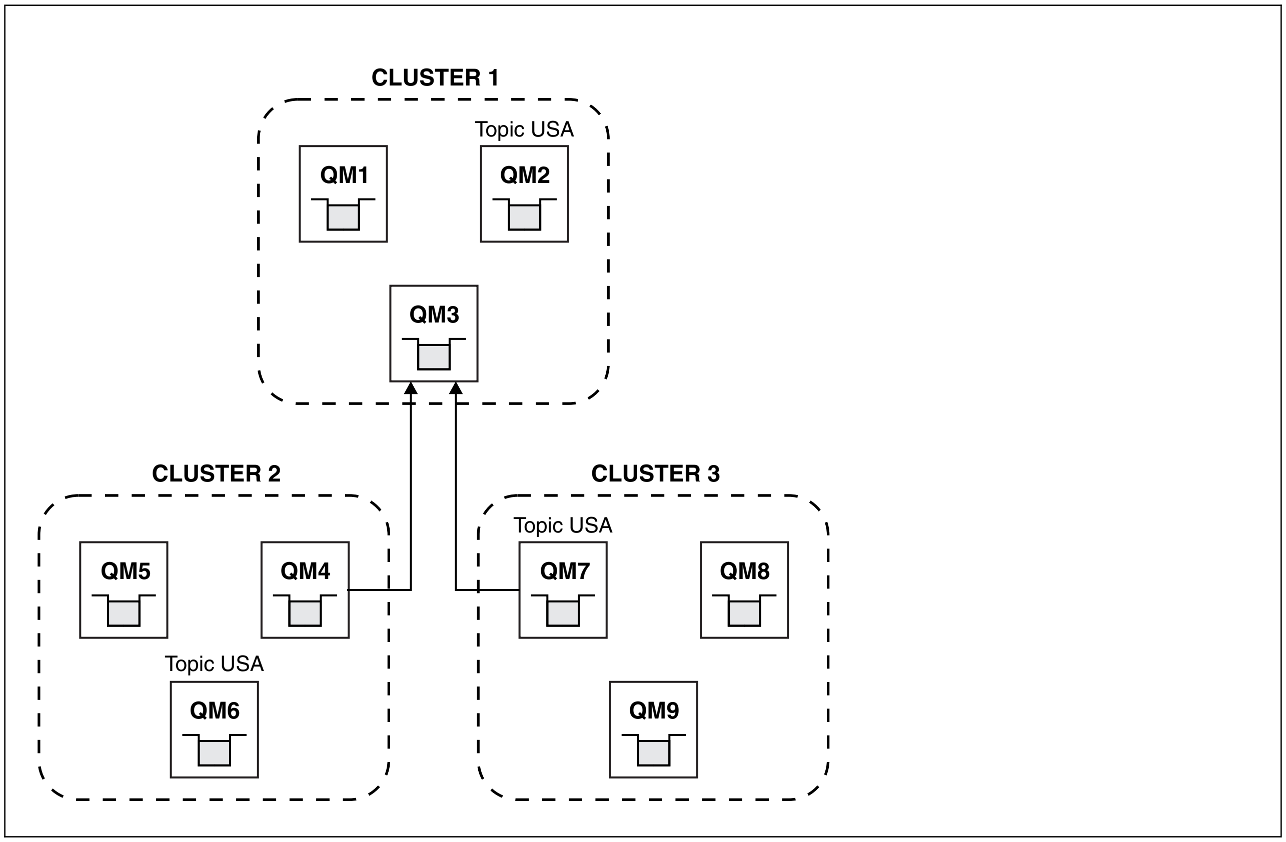


Figure 144. Connecting clusters using hierarchies

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 144. 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."

Related information:

Distributed publish/subscribe networks

Topic spaces

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 1177. 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 144 on page 1178 in "Combining the topic spaces of multiple clusters" on page 1177 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 145 on page 1180. Use the bridging queue manager to filter which publications and subscriptions can flow from one cluster to another.

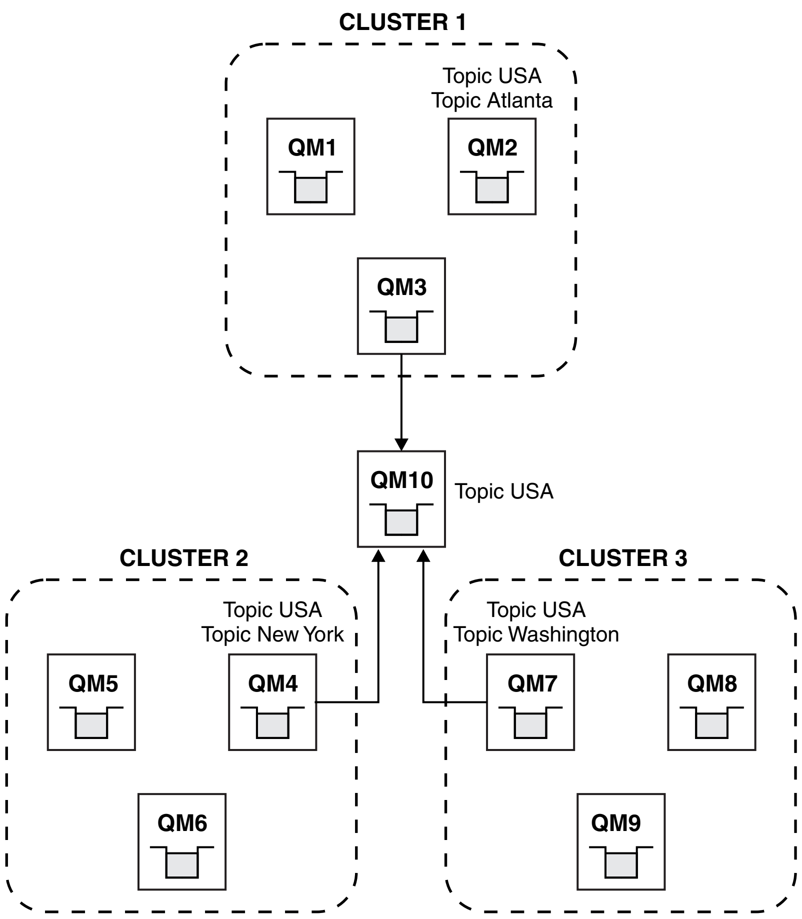


Figure 145. Bridged clusters

Use the bridge to isolate cluster topics that you do not want exposed across the bridge on the other clusters. In Figure 145, `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 145, 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 information:

Distributed publish/subscribe networks

Topic spaces

Defining cluster topics

Publication scope

Subscription scope

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 146 on page 1182. 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.

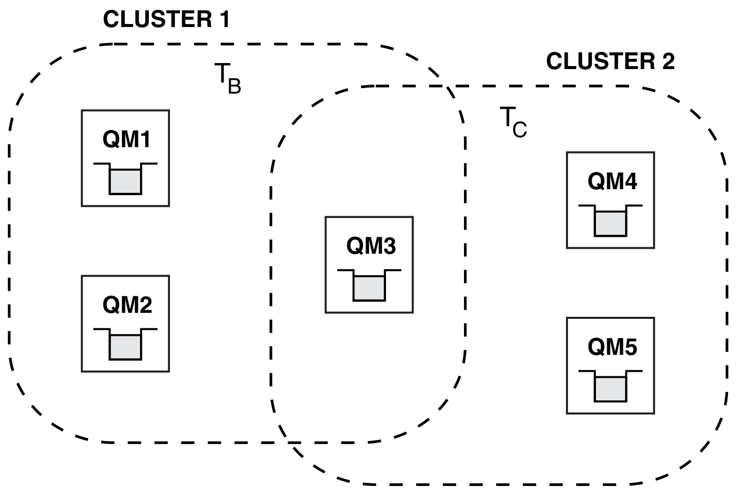


Figure 146. Overlapping clusters, non-overlapping topic spaces

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.

1. `DEFINE TOPIC(B) TOPICSTR('B') CLUSTER('CLUSTER 1') ...`
2. `DEFINE TOPIC(C) TOPICSTR('C') CLUSTER('CLUSTER 2') ...`

Example

In Figure 146 publishers and subscriber connected to `QM3` can publish or subscribe to T_B or T_C

What to do next

Connect publishers and subscribers that use topics in both clusters to queue managers in the overlap.

Connect publishers and subscribers that must only use topics in a specific cluster to queue managers not in the overlap.

Related information:

Distributed publish/subscribe networks

Topic spaces

Defining cluster topics

Connecting a queue manager to a publish/subscribe hierarchy

You connect the child queue manager to the parent queue manager in the hierarchy. If the child queue manager is already a member of another hierarchy or cluster, then this connection joins the hierarchies together, or joins the cluster to the hierarchy.

Before you begin

1. Queue managers in a publish/subscribe hierarchy must have unique queue manager names.
2. A publish/subscribe hierarchy relies on the “queued publish/subscribe” queue manager feature. This must be enabled on both the parent and the child queue managers. See Starting queued publish/subscribe.

3. The publish/subscribe relationship relies on queue manager sender and receiver channels. There are two ways to establish the channels:
   - Add both the parent and child queue managers to a IBM MQ cluster. See Adding a queue manager to a cluster.
   - Establish a sender/receiver channel pair from the child queue manager to the parent and from the parent to the child. Each channel either needs to use a transmission queue with the same name as the target queue manager, or a queue manager alias with the same name as the target queue manager. For more information about how to establish a point-to-point channel connection, see IBM MQ distributed queuing techniques.

   For examples that configure a hierarchy over each type of channel configuration, see the following set of publish/subscribe hierarchy scenarios:
   - Scenario 1: Using point-to-point channels with queue manager name alias
   - Scenario 2: Using point-to-point channels with same name for transmission queue and remote queue manager
   - Scenario 3: Using a cluster channel to add a queue manager

About this task

Use the `ALTER QMGR PARENT (`*`PARENT_NAME`*`)` **`runmqsc`** command to connect children to parents. This configuration is performed on the child queue manager, where *PARENT_NAME* is the name of the parent queue manager.

Procedure

ALTER QMGR PARENT(*PARENT_NAME*)

Example

The first example shows how to attach queue manager QM2 as a child of QM1, then query QM2 to confirm it has successfully become a child with a **`STATUS`** of `ACTIVE`:

```
C:>runmqsc QM2
5724-H72 (C) Copyright IBM Corp. 1994, 2008.  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, 2008.  ALL RIGHTS RESERVED.
Starting MQSC for queue manager QM1.
display pubsub all
     2 : display pubsub all
AMQ8723: Display pub/sub status details.
   QMNAME(QM1)                         TYPE(LOCAL)
   STATUS(ACTIVE)
AMQ8723: Display pub/sub status details.
   QMNAME(QM2)                         TYPE(CHILD)
   STATUS(ACTIVE)
```

If **STATUS** does not show as ACTIVE, check that the channels between the child and the parent are correctly configured and running. Check both queue manager error logs for possible errors.

What to do next

By default, topics used by publishers and subscribers on one queue manager are shared with publishers and subscribers on the other queue managers in the hierarchy. Administered topics can be configured to control the level of sharing through use of the **SUBSCOPE** and **PUBSCOPE** topic properties. See Configuring distributed publish/subscribe networks.

Related information:

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, 2008.  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.

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 UNIX, Linux, and Windows.

Procedure

Use the information in the following links to configure your installations:
- "Changing the primary installation" on page 1194
- "Associating a queue manager with an installation" on page 1196
- "Connecting applications in a multiple installation environment"

Connecting applications in a multiple installation environment

On UNIX, Linux, and Windows systems, if IBM WebSphere MQ Version 7.1, or later, libraries are loaded, IBM MQ automatically uses the appropriate libraries without you needing to take any further action. IBM MQ uses libraries from the installation associated with the queue manager that the application connects to.

The following concepts are used to explain the way applications connect to IBM MQ:

Linking
When the application is compiled, the application is linked to the IBM MQ libraries to get the function exports that are then loaded when the application runs.

Loading
When the application is run, the IBM MQ libraries are located and loaded. The specific mechanism used to locate the libraries varies by operating system, and by how the application is built. For more information about how to locate and load libraries in a multiple installation environment, see "Loading IBM MQ libraries" on page 1187.

Connecting
When the application connects to a running queue manager, for example, using a `MQCONN` or `MQCONNX` call, it connects using the loaded IBM MQ libraries.

When a server application connects to a queue manager, the loaded libraries must come from the installation associated with the queue manager. With multiple installations on a system, this restriction introduces new challenges when choosing the mechanism that the operating system uses to locate the IBM MQ libraries to load:
- When the **`setmqm`** command is used to change the installation associated with a queue manager, the libraries that need to be loaded change.
- When an application connects to multiple queue managers that are owned by different installations, multiple sets of libraries need to be loaded.

However, if IBM WebSphere MQ Version 7.1, or later, libraries, are located and loaded, IBM MQ then loads and uses the appropriate libraries without you needing to take any further action. When the application connects to a queue manager, IBM MQ loads libraries from the installation that the queue manager is associated with.

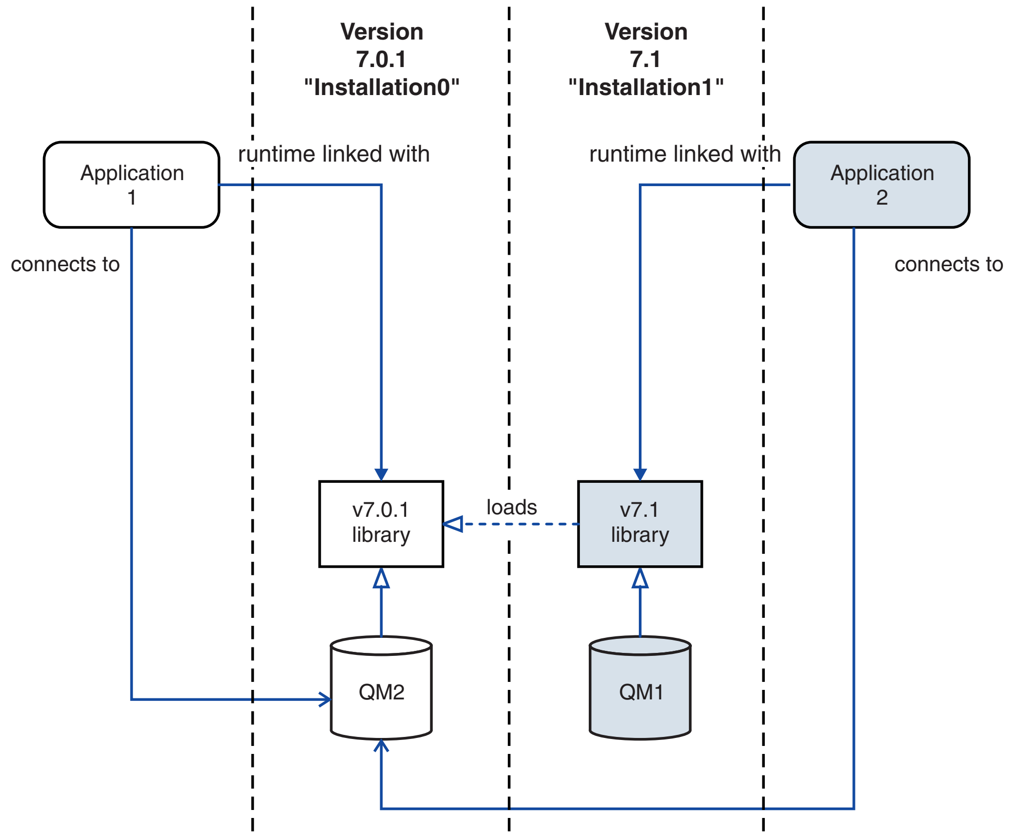


Figure 147. Connecting applications in a multiple installation environment

For example, Figure 147 shows a multiple installation environment with a Version 7.0.1 installation (`Installation0`), and a Version 7.1 installation (`Installation1`). Two applications are connected to these installations, but they load different library versions.

`Application 1` directly loads a Version 7.0.1 library. When `application 1` connects to `QM2`, the Version 7.0.1 libraries are used . If `application 1` attempts to connect to `QM1`, or if `QM2` is associated with `Installation1`, `application 1` fails with a 2059 (080B) (RC2059): MQRC_Q_MGR_NOT_AVAILABLE error. The application fails because the Version 7.0.1 library is not capable of loading other library versions. That is, if Version 7.0.1 libraries are directly loaded, you cannot use a queue manager associated with an installation at a later version of IBM MQ.

`Application 2` directly loads a Version 7.1 library. When `application 2` connects to `QM2`, the Version 7.1 library then loads and uses the Version 7.0.1 library. If `application 2` connects to `QM1`, or if `QM2` is associated with `Installation1`, the Version 7.1 library is loaded, and the application works as expected.

Migration scenarios and connecting applications with multiple installations is considered in more detail in Multi-installation queue manager coexistence on UNIX, Linux, and Windows.

For more information about how to load IBM WebSphere MQ Version 7.1 libraries, see "Loading IBM MQ libraries" on page 1187.

Support and restrictions

If any of the following Version 7.1, or later, libraries, are located and loaded, IBM MQ can automatically load and use the appropriate libraries:
- The C server libraries
- The C++ server libraries
- The XA server libraries
- The COBOL server libraries
- The COM+ server libraries
- .NET in unmanaged mode

IBM MQ also automatically loads and uses the appropriate libraries for Java and JMS applications in bindings mode.

There are a number of restrictions for applications using multiple installations. For more information, see "Restrictions for applications using multiple installations" on page 1191.

Related concepts:

"Restrictions for applications using multiple installations" on page 1191
There are restrictions when using CICS server libraries, fast path connections, message handles, and exits in a multiple installation environment.

"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.

Related tasks:

"Changing the primary installation" on page 1194
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 1196
When you create a queue manager, it is automatically associated with the installation that issued the **crtmqm** command. On UNIX, Linux, and Windows, you can change the installation associated with a queue manager using the **setmqm** command.

Related information:

Choosing a primary installation

Loading IBM MQ libraries

ULW

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.

This information applies to IBM WebSphere MQ Version 7.1, or later version, libraries.

How IBM MQ libraries are located and loaded depends on your installation environment:
- On UNIX and Linux systems, if a copy of IBM WebSphere MQ Version 7.1, or later version, is installed in the default location, existing applications continue to work in the same way as previous versions. However, if the applications need symbolic links in `/usr/lib`, you must either select a Version 7.1, or later version, installation to be the primary installation, or manually create the symbolic links.
- If IBM WebSphere MQ Version 7.1, or later version, is installed in a non-default location, which is the case if IBM WebSphere MQ Version 7.0.1 is also installed, 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 1190.

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 124. Benefits and drawbacks of the options for loading libraries

Platform	Option	Benefits	Drawbacks
UNIX Linux UNIX and Linux systems	Set or change the embedded runtime search path (RPath) of the application. This option requires you to recompile and link the application. For more information about compiling and linking applications, see Building a procedural application.	• Scope of the change is clear.	• You must be able to recompile and link the application. • If the location of IBM MQ changes, you must change the RPath.
UNIX and Linux systems	Set the *LD_LIBRARY_PATH* environment variable , using setmqenv, or crtmqenv, with the **-k** or **-l** option. (AIX On AIX, this environment variable is *LIBPATH*	• No changes to existing applications required. • Overrides embedded RPaths in an application. • Easy to change the variable if the location of IBM MQ changes.	• `setuid` and `setgid` applications, or applications built in other ways, might ignore *LD_LIBRARY_PATH* for security reasons. • Environment specific, so must be set in each environment where the application is run. • Possible impact on other applications that rely on *LD_LIBRARY_PATH*. • HP-UX HP-UX: Options used when the application was compiled might disable the use of *LD_LIBRARY_PATH*. For more information, see Runtime linking considerations for HP-UX. • Linux Linux: The compiler used to build the application might disable the use of *LD_LIBRARY_PATH*. For more information, see Runtime linking considerations for Linux.

Table 124. Benefits and drawbacks of the options for loading libraries (continued)

Platform	Option	Benefits	Drawbacks
Windows Windows systems	Set the PATH variable using setmqenv, or crtmqenv.	• No changes required for existing applications. • Easy to change the variable if the location of IBM MQ changes.	• Environment specific, so must be set in each environment where the application is run. • Possible impact on other applications.
ULW UNIX, Linux, and Windows systems	Set the primary installation to a Version 7.1, or later, installation. See "Changing the primary installation" on page 1194. For more information about the primary installation, see Choosing a primary installation.	• No changes required for existing applications. • Easy to change the primary installation if the location of IBM MQ changes. • Gives similar behavior to previous versions of IBM MQ.	• When IBM WebSphere MQ Version 7.0.1 is installed, you cannot set the primary installation to Version 7.1, or later. • UNIX Linux UNIX and Linux: Does not work if `/usr/lib` is not in the default search path.

HP-UX

Library loading considerations for HP-UX

The sample compilation commands in the product documentation for previous versions of IBM MQ included the `-Wl, +noenvvar` link option for 64-bit applications. This option disables the use of *LD_LIBRARY_PATH* to load shared libraries. If you want your applications to load IBM MQ libraries from a location other than the location specified in the RPath, you must update your applications. You can update the applications by recompiling and linking without the `-Wl, +noenvvar` link option, or by using the **`chatr`** command.

To find out how your applications currently load libraries, see "Operating system library loading mechanisms" on page 1190.

Linux

Library loading considerations for Linux

Applications compiled using some versions of gcc, for example, version 3.2.x, can have an embedded RPath that cannot be overridden using the *LD_LIBRARY_PATH* environment variable. You can determine if an application is affected by using the `readelf -d` *`applicationName`* command. The RPath cannot be overridden if the RPATH symbol is present and the RUNPATH symbol is not present.

Solaris

Library loading considerations for Solaris

The sample compilation commands in the product documentation for previous versions of IBM MQ included the `-lmqmcs -lmqmzse` link options. The appropriate versions of these libraries are now loaded automatically by IBM MQ. If IBM MQ is installed in a non-default location, or if there are multiple installations on the system, you must update your applications. You can update the applications by recompiling and linking without the `-lmqmcs -lmqmzse` link options.

Operating system library loading mechanisms

On Windows systems, several directories are searched to find the libraries:

- The directory the application is loaded from.
- The current directory.
- The directories in the *PATH* environment variable, both the global *PATH* variable and the *PATH* variable of the current user.

UNIX Linux On UNIX and Linux systems, there are a number of methods that might have been used to locate the libraries to load:

- Using the *LD_LIBRARY_PATH* environment variable (also *LIBPATH* on AIX, and *SHLIB_PATH* on HP-UX). If this variable is set, it defines a set of directories that are searched for the required IBM MQ libraries. If any libraries are found in these directories, they are used in preference of any libraries that might be found using the other methods.
- Using an embedded search path (RPath). The application might contain a set of directories to search for the IBM MQ libraries. If the *LD_LIBRARY_PATH* is not set, or if the required libraries were not found using the variable, the RPath is searched for the libraries. If your existing applications use an RPath, but you cannot recompile and link the application, you must either install IBM WebSphere MQ Version 7.1 in the default location, or use another method to find the libraries.
- Using the default library path. If the IBM MQ libraries are not found after searching the *LD_LIBRARY_PATH* variable and RPath locations, the default library path is searched. Usually, this path contains `/usr/lib` or `/usr/lib64`. If the libraries are not found after searching the default library path, the application fails to start because of missing dependencies.

You can use operating system mechanisms to find out if your applications have an embedded search path. For example:

- AIX AIX: **dump**
- HP-UX HP-UX: **chatr**
- Linux Linux: **readelf**
- Solaris Solaris: **elfdump**

Related concepts:

"Restrictions for applications using multiple installations" on page 1191
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 1185
On UNIX, Linux, and Windows systems, if IBM WebSphere MQ Version 7.1, or later, libraries are loaded, IBM MQ automatically uses the appropriate libraries without you needing to take any further action. IBM MQ uses libraries from the installation associated with the queue manager that the application connects to.

Related tasks:

"Changing the primary installation" on page 1194
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 1196
When you create a queue manager, it is automatically associated with the installation that issued the **crtmqm** command. On UNIX, Linux, and Windows, you can change the installation associated with a queue manager using the **setmqm** command.

Related information:

Choosing a primary installation

Restrictions for applications using multiple installations

ULW

There are restrictions when using CICS server libraries, fast path connections, message handles, and exits in a multiple installation environment.

CICS server libraries

If you are using the CICS server libraries, IBM MQ does not automatically select the correct library level for you. You must compile and link your applications with the appropriate library level for the queue manager to which the application connects. For more information, see Building libraries for use with TXSeries for Multiplatforms version 5.

Message handles

Message handles that use the special value of `MQHC_UNASSOCIATED_HCONN` are limited to use with the first installation loaded in a process. If the message handle cannot be used by a particular installation, reason code `MQRC_HMSG_NOT_AVAILABLE` is returned.

This restriction affects message properties. You cannot use message handles to get message properties from a queue manager on one installation and put them to a queue manager on a different installation. For more information about message handles, see MQCRTMH - Create message handle.

Exits

In a multiple installation environment, existing exits must be updated for use with IBM WebSphere MQ Version 7.1, or later, installations. Data conversion exits generated using the **crtmqcvx** command must be regenerated using the updated command.

All exits must be written using the `MQIEP` structure, cannot use an embedded `RPATH` to locate the IBM MQ libraries, and cannot link to the IBM MQ libraries. For more information, see Writing exits and installable services on UNIX, Linux, and Windows .

Fast path

On a server with multiple installations, applications using a fast path connection to IBM WebSphere MQ Version 7.1 or later must follow these rules:

1. The queue manager must be associated with the same installation as the one from which the application loaded the IBM MQ run time libraries. The application must not use a fast path connection to a queue manager associated with a different installation. An attempt to make the connection results in an error, and reason code `MQRC_INSTALLATION_MISMATCH`.
2. Connecting non-fast path to a queue manager associated with the same installation as the one from which the application has loaded the IBM MQ run time libraries prevents the application connecting fast path, unless either of these conditions are true:
   - The application makes its first connection to a queue manager associated with the same installation a fast path connection.
   - The environment variable, `AMQ_SINGLE_INSTALLATION` is set.
3. Connecting non-fast path to a queue manager associated with a Version 7.1 or later installation, has no effect on whether an application can connect fast path.
4. You cannot combine connecting to a queue manager associated with a Version 7.0.1 installation and connecting fast path to a queue manager associated with a Version 7.1, or later installation.

With `AMQ_SINGLE_INSTALLATION` set, you can make any connection to a queue manager a fast path connection. Otherwise almost the same restrictions apply:

- The installation must be the same one from which the IBM MQ run time libraries were loaded.
- Every connection on the same process must be to the same installation. If you attempt to connect to a queue manager associated with a different installation, the connection fails with reason code `MQRC_INSTALLATION_MISMATCH`. Note that with `AMQ_SINGLE_INSTALLATION` set, this restriction applies to all connections, not only fast path connections.
- Only connect one queue manager with fast path connections.

Related information:

MQCONNX - Connect queue manager (extended)

MQIEP structure

2583 (0A17) (RC2583): MQRC_INSTALLATION_MISMATCH

2587 (0A1B) (RC2587): MQRC_HMSG_NOT_AVAILABLE

2590 (0A1E) (RC2590): MQRC_FASTPATH_NOT_AVAILABLE

Connecting .NET applications in a multiple installation environment

ULW

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.”

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 1193.

Connecting .NET applications using the application configuration file

Within the application configuration file, you must set various tags to redirect applications to use assemblies that are not from the primary installation.

The following table shows the specific changes that need to be made to the application configuration file to allow .NET applications connect using particular assemblies:

Table 125. Configuring applications to use particular assemblies

	Applications compiled with an earlier version of IBM MQ	**Applications compiled with a later version of IBM MQ**
To run an application with a later version IBM MQ primary installation. (later version assemblies in GAC):	No changes necessary	No changes necessary
To run an application with an earlier version IBM MQ primary installation. (earlier version assemblies in GAC):	No changes necessary	In the application configuration file: • Use the *bindingRedirect* tag to indicate the use of the earlier version of the assemblies that are in the GAC
To run an application with a later version of IBM MQ non-primary installation. (later version assemblies in installation folder):	In the application configuration file: • Use the *codebase* tag to point to the location of the later version assemblies • Use the *bindingRedirect* tag to indicate the use of the later version assemblies	In the application configuration file: • Use the *codebase* tag to point to the location of the later version assemblies
To run an application with an earlier version of IBM MQ non-primary installation. (earlier version assemblies in installation folder):	In the application configuration file: • Use the *codebase* tag to point to the location of the earlier version assemblies • Include the tag *publisherpolicy Apply=no*	In the application configuration file: • Use the *codebase* tag to point to the location of the earlier version assemblies • Use the *bindingRedirect* tag to indicate the use of the earlier version assemblies • Include the tag *publisherpolicy Apply=no*

A sample application configuration file `NonPrimaryRedirect.config` is shipped in the folder *`MQ_INSTALLATION_PATH`*`\tools\dotnet\samples\base`. This file can be modified with the IBM MQ installation path of any non-primary installation. The file can also be directly included in other configuration files using the *linkedConfiguration* tag. Samples are provided for `nmqsget.exe.config` and `nmqsput.exe.config`. Both samples use the *linkedConfiguration* tag and include the `NonPrimaryRedirect.config` file.

Connecting .NET applications using DEVPATH

You can find the assemblies using the *DEVPATH* environment variable. The assemblies specified by the *DEVPATH* variable are used in preference to any assemblies in the GAC. See the appropriate Microsoft documentation on *DEVPATH* for more information about when to use this variable.

To find the assemblies using the *DEVPATH* environment variable, you must set the *DEVPATH* variable to the folder that contains the assemblies you want to use. Then, you must then update the application configuration file and add the following runtime configuration information:

```
<configuration>
<runtime>
<developmentMode developerInstallation="true" />
</runtime>
</configuration>
```

Related concepts:

"Connecting applications in a multiple installation environment" on page 1185
On UNIX, Linux, and Windows systems, if IBM WebSphere MQ Version 7.1, or later, libraries are loaded, IBM MQ automatically uses the appropriate libraries without you needing to take any further action. IBM MQ uses libraries from the installation associated with the queue manager that the application connects to.

Related information:

Choosing a primary installation

Using .NET

Multiple installations

Changing the primary installation

ULW

You can use the **setmqinst** command to set or unset an installation as the primary installation.

About this task

This task applies to UNIX, Linux, and Windows.

The primary installation is the installation to which required system-wide locations refer. For more information about the primary installation, and considerations for choosing your primary installation, see Choosing a primary installation.

If an installation of IBM WebSphere MQ Version 7.1 or later is coexisting with an installation of IBM WebSphere MQ Version 7.0.1, the IBM WebSphere MQ Version 7.0.1 installation must be the primary. It is flagged as primary when the IBM WebSphere MQ Version 7.1 or later version is installed, and the IBM WebSphere MQ Version 7.1 or later installation cannot be made primary.

Windows During the installation process on Windows, you can specify that the installation is to be the primary installation.

UNIX Linux On UNIX and Linux systems, you must issue a **setmqinst** command after installation to set the installation as the primary installation.

"Set the primary installation" on page 1195.

"Unset the primary installation" on page 1195.

Set the primary installation

Procedure

To set an installation as the primary installation:

1. Check if an installation is already the primary installation by entering the following command:

   ```
   MQ_INSTALLATION_PATH/bin/dspmqinst
   ```

   where *MQ_INSTALLATION_PATH* is the installation path of a IBM WebSphere MQ Version 7.1 or later installation.
2. If an existing IBM WebSphere MQ Version 7.1 or later installation is set as the primary installation, unset it by following the instructions in "Unset the primary installation." If IBM WebSphere MQ Version 7.0.1 is installed on the system, the primary installation cannot be changed.
3. Make sure that you are logged on with the appropriate authority:
   - UNIX As root on UNIX and Linux.
   - Linux As a member of the Administrators group on Windows systems.
4. 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
     ```
5. Windows On Windows systems, restart the system.

Unset the primary installation

Procedure

To unset an installation as the primary installation:

1. Check which installation is the primary installation by entering the following command:

   ```
   MQ_INSTALLATION_PATH/bin/dspmqinst
   ```

   where *MQ_INSTALLATION_PATH* is the installation path of a IBM WebSphere MQ Version 7.1 or later installation.

   If IBM WebSphere MQ Version 7.0.1 is the primary installation, you cannot unset the primary installation.
2. Make sure that you are logged on with the appropriate authority:
   - UNIX As root on UNIX and Linux.
   - Linux As a member of the Administrators group on Windows systems.
3. 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 information:

Features that can be used only with the primary installation on Windows

External library and control command links to primary installation on UNIX and Linux

Uninstalling, upgrading, and maintaining the primary installation

Choosing an installation name

setmqinst

Associating a queue manager with an installation

ULW

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

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** usually has to be issued from the installation that is associated with the queue manager. However, when a Version 7.0.1 or earlier queue manager is started on a Version 7.1 or later installation for the first time, **strmqm** can be used. In this case, **strmqm** starts the queue manager and associates it with the installation from which the command is issued.
- **dspmq** displays information about all queue managers on a system, not just those queue managers associated with the same installation as the **dspmq** command. The `dspmq -o installation` command displays information about which queue managers are associated with which installations.

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 **dltmqm** command from that installation to delete the queue manager.

Related concepts:

"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.

"IBM MQ configuration file, mqs.ini" on page 909
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 information:

Choosing a primary installation

addmqinf

dspmq

dspmqinst

endmqm

setmqm

strmqm

Finding installations of IBM MQ on a system

ULW

If you have multiple IBM MQ installations on a system, you can check which versions are installed and where they are.

You can use the following methods to find the IBM MQ installations on your system:

- Use the **dspmqver** command. This command does not provide details of all installations on a system if it is issued from a Version 7.0.1 installation.
- Use the platform installation tools to query where IBM MQ has been installed. Then use the **dspmqver** command from a Version 7.1 or later installation. The following commands are examples of commands you can use to query where IBM MQ has been installed:
  - On AIX systems, you can use the **lslpp** command:
    ```
    lslpp -R ALL -l mqm.base.runtime
    ```
  - On HP-UX systems, you can use the **swlist** command:
    ```
    swlist -a location -a revision -l product MQSERIES
    ```
  - On Linux systems, you can use the **rpm** command:
    ```
    rpm -qa --qf "%{NAME}-%{VERSION}-%{RELEASE}\t%{INSTPREFIXES}\n" | grep MQSeriesRuntime
    ```
  - On Solaris systems, you can use the **pkginfo** and **pkgparam** commands:

1. List the installed packages by entering the following command:
   ```
   pkginfo | grep -w mqm
   ```
2. For each package listed, enter following command:
   ```
   pkgparam pkgname BASEDIR
   ```

– On Windows systems, you can use the **wmic** command. This command might install the wmic client:
  ```
  wmic product where "(Name like '%MQ%') AND (not Name like '%bitSupport')" get Name, Version, InstallLocation
  ```

- On UNIX and Linux systems, issue the following command to find out where IBM MQ has been installed:
  ```
  cat /etc/opt/mqm/mqinst.ini
  ```
  Then use the **dspmqver** command from a Version 7.1 or later installation.
- To display details of installations on the system, on 32-bit Windows, issue the following command:
  ```
  reg.exe query "HKEY_LOCAL_MACHINE\SOFTWARE\IBM\WebSphere MQ\Installation" /s
  ```
- On 64-bit Windows, issue the following command:
  ```
  reg.exe query "HKEY_LOCAL_MACHINE\SOFTWARE\Wow6432Node\IBM\WebSphere MQ\Installation" /s
  ```

Note: the **reg.exe** command will only display information for Version 7.1 or later installations.

Related information:

dspmqver

dspmqinst

Multiple installations

Configuring high availability, recovery and restart

You can make your applications highly available by maintaining queue availability if a queue manager fails, and by recovering messages after server or storage failure.

About this task

z/OS On z/OS, high availability is built into the platform. Extra servant regions are spawned as needed, to meet increased demand. You can also improve server application availability by using queue-sharing groups. See Shared queues and queue-sharing groups.

Multi On Multiplatforms, you can improve client application availability by using client reconnection to switch a client automatically between a group of queue managers, or to the new active instance of a multi-instance queue manager after a queue manager failure. Automatic client reconnect is not supported by IBM MQ classes for Java. A multi-instance queue manager is configured to run as a single queue manager on multiple servers. You deploy server applications to this queue manager. If the server running the active instance fails, execution is automatically switched to a standby instance of the same queue manager on a different server. If you configure server applications to run as queue manager services, they are restarted when a standby instance becomes the actively running queue manager instance.

Another way to increase server application availability on Multiplatforms is to deploy server applications to multiple computers in a queue manager cluster. From IBM WebSphere MQ Version 7.1 onwards, cluster error recovery reruns operations that caused problems until the problems are resolved. See Changes to cluster error recovery on servers other than z/OS. You can also configure IBM MQ for Multiplatforms as part of a platform-specific clustering solution such as:
- Microsoft Cluster Server
- IBM i HA clusters on IBM i

- Linux UNIX PowerHA® for AIX (formerly HACMP™ on AIX) and other UNIX and Linux clustering solutions

A messaging system ensures that messages entered into the system are delivered to their destination. IBM MQ can trace the route of a message as it moves from one queue manager to another using the **dspmqrte** command. If a system fails, messages can be recovered in various ways depending on the type of failure, and the way a system is configured. IBM MQ 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.

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 126 on page 1200 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 127 on page 1200 lists other requirements to enable automatic client reconnection.

With program access to reconnection options, a client application can set reconnection options. Except for JMS and XMS clients, if a client application has access to reconnection options, it can also create an event handler to handle reconnection events.

An existing client application might be able to benefit from reconnection support, without recompilation and linking:

- For a non-JMS client, set the `mqclient.ini` environment variable `DefRecon` to set reconnection options. Use a CCDT to connect to a queue manager. If the client is to connect to a multi-instance queue manager, provide the network addresses of the active and standby queue manager instances in the CCDT.
- For a JMS client, set the reconnection options in the connection factory configuration. When running inside the EJB container of a Java EE server, MDBs can reconnect to IBM MQ using the reconnect mechanism provided by activation specifications of the IBM MQ resource adapter (or listener ports if running in WebSphere Application Server). However, if the application is not an MDB (or is running in the web container) the application must implement its own reconnect logic because automatic client reconnect is not supported in this scenario. The IBM MQ resource adapter provides this reconnect ability for the delivery of messages to message driven beans, but other Java EE elements such as servlets must implement their own reconnection.

Table 126. Supported clients

Client interface	Client	Program access to reconnection options	Reconnection support
Messaging APIs	C, C++, COBOL, Unmanaged Visual Basic, XMS (Unmanaged XMS on Windows)	7.0.1	7.0.1
	JMS (JSE, and Java EE client container and managed containers)	7.0.1.3	7.0.1.3
	IBM MQ classes for Java	Not supported	Not supported
	Managed XMS and managed .NET clients: C#, Visual Basic,	7.1	7.1
Other APIs	Windows Communication Foundation (Unmanaged [1])	Not supported	7.0.1
	Windows Communication Foundation (Managed [1])	Not supported	Not supported
	Axis 1	Not supported	Not supported
	Axis 2	Not supported	7.0.1.3
	HTTP (web 2.0)	Not supported	7.0.1.3

1. Set managed or unmanaged mode in the WCF binding configuration.

Automatic reconnection has the following configuration requirements:

Table 127. Automatic reconnection configuration requirements

Component	Requirement	Effect of not meeting requirement
IBM MQ MQI client installation	See Table 126	`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: • MQCONNX with `MQCNO Options` set to `MQCNO_RECONNECT` or `MQCNO_RECONNECT_Q_MGR`. • `Defrecon=YES|QMGR` in `mqclient.ini` • In JMS set the CLIENTRECONNECTOPTIONS property of the connection factory.	`MQCC_FAILED` when a connection is broken or queue manager ends or fails.

Figure 148 on page 1201 shows the main interactions between components that are involved in client reconnection.

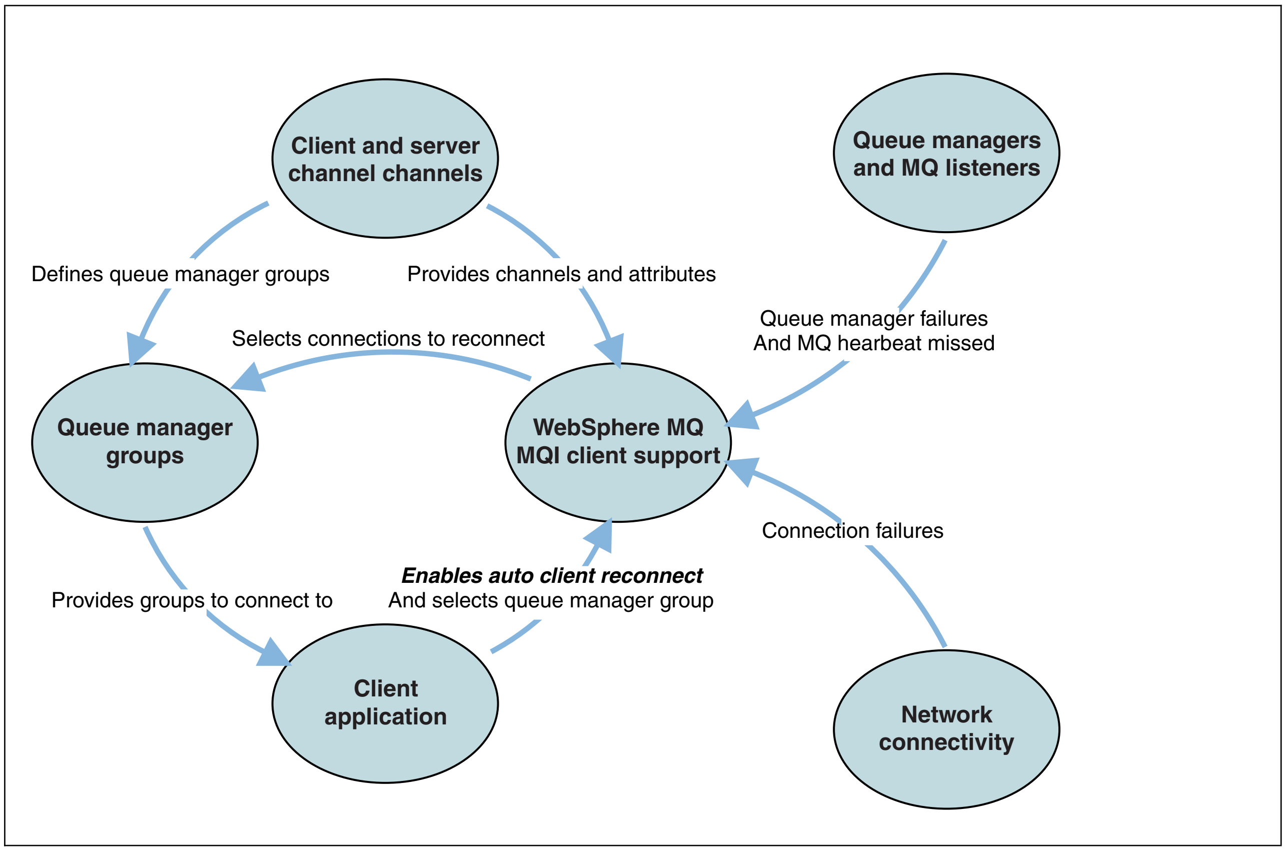

Client and server channel channels
Queue managers and MQ listeners
Defines queue manager groups
Provides channels and attributes
Queue manager failures And MQ hearbeat missed
Selects connections to reconnect
Queue manager groups
WebSphere MQ MQI client support
Connection failures
Provides groups to connect to
Enables auto client reconnect And selects queue manager group
Client application
Network connectivity

Figure 148. Automatic client reconnection

Client application

The client application is an IBM MQ MQI client.

- By default clients are not automatically reconnected. Enable the automatic client reconnection by setting the MQCONNX `MQCNO Option MQCNO_RECONNECT` or `MQCNO_RECONNECT_Q_MGR`.
- Many applications are written in such a way that they are able to take advantage of auto-reconnection with no additional coding. Enable automatic reconnection for existing programs, without making any coding changes, by setting the `DefRecon` attribute in the channels stanza of the `mqclient.ini` configuration file.
- Use one of these three options:
  1. Modify the program so that the logic is unaffected by reconnection. For example, you might have to issue MQI calls within the sync point, and resubmit backed-out transactions.
  2. Add an event handler to detect reconnection, and restore the state of the client application when the connection is reestablished.
  3. Do not enable auto-reconnection: instead, disconnect the client and issue a new MQCONN or MQCONNX MQI call to find another queue manager instance that is running in the same queue manager group.

  For further details about these three options, see "Application recovery" on page 1286.
- Reconnecting to a queue manager of the same name does not guarantee that you have reconnected to the same instance of a queue manager.

  Use an `MQCNO` option `MQCNO_RECONNECT_Q_MGR`, to reconnect to an instance of the same queue manager.
- A client can register an event handler so that it can be informed the state of reconnection. The `MQHCONN` passed in the event handler cannot be used. The following reason codes are provided:

MQRC_RECONNECTING
: The connection failed, and the system is attempting to reconnect. You receive multiple `MQRC_RECONNECTING` events if multiple reconnect attempts are made.

MQRC_RECONNECTED
: The reconnection made and all handles successfully reestablished.

MQRC_RECONNECT_FAILED
: The reconnection was not successful.

MQRC_RECONNECT_QMID_MISMATCH
: A reconnectable connection specified `MQCNO_RECONNECT_Q_MGR` and the connection attempted to reconnect to a different queue manager.

MQRC_RECONNECT_Q_MGR_REQD
: An option, such `MQMO_MATCH_MSG_TOKEN` in an MQGET call, was specified in the client program that requires reconnection to the same queue manager.

- A reconnectable client is able to reconnect automatically only *after* connecting. That is, the MQCONNX call itself is not tried again if it fails. For example, if you receive the return code `2543 - MQRC_STANDBY_Q_MGR` from MQCONNX, reissue the call after a short delay.

MQRC_RECONNECT_INCOMPATIBLE
: This reason code is returned when the application tries to use `MQPMO_LOGICAL_ORDER` (with MQPUT and MQPUT1) or `MQGMO_LOGICAL_ORDER` (with MQGET) when reconnect options are set. The reason for returning the reason code is to make sure that applications never use reconnect in such cases.

MQRC_CALL_INTERRUPTED
: This reason code is returned when the connection breaks during the execution of Commit call and the client reconnects. An MQPUT of a persistent message outside the sync point also results in the same reason code being returned to the application.

Multi-instance queue managers

Simplify restarting IBM MQ MQI client applications, after a multi-instance queue manager has activated its standby instance, by using automatic client reconnection.

The standby instance of a multi-instance queue manager is typically at a different network address to the active instance. Include the network addresses of both the instances in the client connection definition table (CCDT). Either provide a list of network addresses for the **`CONNAME`** parameter, or define multiple rows for the queue manager in the CCDT.

Commonly, IBM MQ MQI clients reconnect to any queue manager in a queue manager group. Sometimes you want an IBM MQ MQI client to reconnect only to the same queue manager. It might have an affinity to a queue manager. You can prevent a client from reconnecting to a different queue manager. Set the `MQCNO` option, `MQCNO_RECONNECT_Q_MGR`. The IBM MQ MQI client fails if it reconnects to a different queue manager. If you set the `MQCNO` option, `MQCNO_RECONNECT_Q_MGR`, do not include other queue managers in the same queue manager group. The client returns an error if the queue manager it reconnects to is not the same queue manager as the one it connected to.

Queue manager groups

You can select whether the client application always connects and reconnects to a queue manager of the same name, to the same queue manager, or to any of a set of queue managers that are defined with the same `QMNAME` value in the client connection table.

- The queue manager *name* attribute, `QMNAME`, in the client channel definition is the name of a queue manager group.

- In your client application, if you set the value of the MQCONN or MQCONNX `QmgrName` parameter to a queue manager name, the client connects only to queue managers with that name. If you prefix the queue manager name with an asterisk(*), the client connects to any queue manager in the queue manager group with the same `QMNAME` value. For a full explanation, see Queue manager groups in the CCDT.

z/OS

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.

  z/OS 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.

  z/OS Note that `KAINT` applies to z/OS only.

`KAINT` is a network layer heartbeat, and `HBINT` is an IBM MQ heartbeat between the client and the queue manager. Setting these heartbeats to a shorter time serves two purposes:

1. By simulating activity on the connection, network layer software that is responsible for closing inactive connections is less likely to shut down your connection.
2. If the connection is shut down, the delay before the broken connection is detected, is shortened.

The default TCP/IP keepalive interval is two hours. Consider setting the `KAINT` and `HBINT` attributes to a shorter time. Do not assume that the normal behavior of a network suits the needs of automatic reconnection. For example, some firewalls can shut down an inactive TCP/IP connection after as little as 10 minutes.

Network connectivity

Only network failures that are passed to the IBM MQ MQI client by the network, are handled by the automatic reconnection capability of the client.

- Reconnections performed automatically by the transport are invisible to IBM MQ.
- Setting `HBINT` helps to deal with network failures that are invisible to IBM MQ.

Queue managers and IBM MQ listeners

Client reconnection is triggered by server failure, queue manager failure, network connectivity failure, and by an administrator switching over to another queue manager instance.

- If you are using a multi-instance queue manager, an additional cause of client reconnection occurs when you switch control from the active queue manager instance to a standby instance.
- Ending a queue manager using the default **endmqm** command, does not trigger automatic client reconnection. Add the `-r` option on the **endmqm** command to request automatic client reconnection, or the `-s` option to transfer to a standby queue manager instance after shutting down.

IBM MQ MQI client automatic reconnection support

If you use the automatic client reconnection support in the IBM MQ MQI client, the client application automatically reconnects and continues processing without you issuing an MQCONN or MQCONNX MQI call to reconnect to the queue manager.

- Automatic client reconnection is triggered by one of the following occurrences:
  - queue manager failure
  - ending a queue manager and specifying the `-r`, reconnect, option on the **endmqm** command
- The MQCONNX MQCNO options control whether you have enabled the automatic client reconnection. The options are described in Reconnection options.
- Automatic client reconnection issues MQI calls on behalf of your application to restore the connection handle and the handles to other open objects, so that your program can resume normal processing after it has processed any MQI errors that resulted from the broken connection. See “Recovery of an automatically reconnected client” on page 1288.
- 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.

Console message monitoring

z/OS

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.

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.

.

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

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.

CSQJ004I: ACTIVE LOG COPY *n* **INACTIVE, LOG IN SINGLE MODE, ENDRBA=** *ttt*

- Allocate enough units to enable archiving.
- Check the queue manager joblog for additional error messages.

.

CSQJ136I: UNABLE TO ALLOCATE TAPE UNIT FOR CONNECTION-ID= *xxxx* **CORRELATION-ID=** *yyyyyy***,** *m* **ALLOCATED** *n* **ALLOWED**

- Check the queue manager joblog for additional error messages.

.

CSQJ151I: *csect-name* **ERROR READING RBA** *rrr***, CONNECTION-ID=** *xxxx* **CORRELATION-ID=** *yyyyyy* **REASON CODE=** *ccc*

- Check the queue manager joblog for additional messages.
- Issue a **DISPLAY CONN** command to determine which connection is not committing its activity.
- Ensure the application can commit its updates.

CSQJ160I: LONG-RUNNING UOW FOUND, URID= *urid* **CONNECTION NAME=** *name*

- Check the queue manager joblog for additional messages.
- Issue a **DISPLAY CONN** command to determine which connection is not committing its activity.
- Ensure the application can commit its updates.

CSQJ161I: UOW UNRESOLVED AFTER *n* **OFFLOADS, URID=** *urid* **CONNECTION NAME=** *name*

- Determine if the page set is allocated to the queue manager.
- Issue a **DISPLAY USAGE** command to determine the state of the page set.
- Check the queue manager joblog for additional messages.

CSQP011E: CONNECT ERROR STATUS *ret-code* **FOR PAGE SET** *psid*

- Check the curdepth of the queues allocated to this page set.
- Investigate the cause of the failure to process messages.

CSQP013I: *csect-name* **NEW EXTENT CREATED FOR PAGE SET psid. NEW EXTENT WILL NOW BE FORMATTED**

- Check the curdepth of the queues allocated to this page set.
- Investigate the cause of failure to process messages.
- Determine if queues need to be relocated to another page set.
- If the volume is full, determine if you need to make the page set a multi volume 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*

- Check the IMS log to determine why the Units of Work are still indoubt.
- 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* **Unable to convert message for** *name***, MQCC=** *mqcc* **MQRC=** *mqrc* **(***mqrc-text***)**

- Check the joblog to determine the cause of the TCP/IP failure.
- Check the TCP/IP address space for errors.

CSQX234I: *csect-name* **Listener stopped, TRPTYPE=** *trptype* **INDISP=** *disposition*

- If the listener does not stop, following a **`STOP`** command, check the TCP/IP address space for errors.
- Follow the 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.

High availability configurations

If you want to operate your IBM MQ queue managers in a high availability (HA) configuration, you can set up your queue managers to work either with a high availability manager, such as PowerHA for AIX (formerly HACMP) or the Microsoft Cluster Service (MSCS), or with IBM MQ multi-instance queue managers. V 9.0.4 On Linux systems, you can also deploy replicated data queue managers (RDQMs), which use a quorum-based group to provide high availability.

You need to be aware of the following configuration definitions:

Queue manager clusters
: Groups of two or more queue managers on one or more computers, providing automatic interconnection, and allowing queues to be shared among them for load balancing and redundancy. From IBM WebSphere MQ Version 7.1 onwards, cluster error recovery reruns operations that caused problems until the problems are resolved.

HA clusters
: HA clusters are groups of two or more computers and resources such as disks and networks, connected together and configured in such a way that, if one fails, a high availability manager, such as HACMP (UNIX) or MSCS (Windows) performs a *failover*. The failover transfers the state data of applications from the failing computer to another computer in the cluster and re-initiates their operation there. This provides high availability of services running within the HA cluster. The relationship between IBM MQ clusters and HA clusters is described in "Relationship of HA clusters to queue manager clusters" on page 1209.

Multi-instance queue managers
: Instances of the same queue manager configured on two or more computers. By starting multiple instances, one instance becomes the active instance and the other instances become standbys. If the active instance fails, a standby instance running on a different computer automatically takes over. You can use multi-instance queue managers to configure your own highly available messaging systems based on IBM MQ, without requiring a cluster technology such as HACMP or MSCS. HA clusters and multi-instance queue managers are alternative ways of making queue managers highly available. Do not combine them by putting a multi-instance queue manager in an HA cluster.

V 9.0.4 **High availability replicated data queue managers (HA RDQMs)**
: Instances of the same queue manager configured on each node in a group of three Linux servers. 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. The grouping of the servers is controlled by Pacemaker, and the replication by DRBD.

V 9.0.5 **Disaster recovery replicated data queue managers (DR RDQMs)**
: 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. Both nodes must be Linux servers. The replication is controlled by DRBD.

Differences between multi-instance queue managers and HA clusters

Multi-instance queue managers and HA clusters are alternative ways to achieve high availability for your queue managers. Here are some points that highlight the differences between the two approaches.

Multi-instance queue managers include the following features:

- Basic failover support integrated into IBM MQ
- Faster failover than HA cluster

- Simple configuration and operation
- Integration with IBM MQ Explorer

Limitations of multi-instance queue managers include:
- Highly available, high performance networked storage required
- More complex network configuration because queue manager changes IP address when it fails over

HA clusters include the following features:
- The ability to coordinate multiple resources, such as an application server or database
- More flexible configuration options including clusters comprising more than two nodes
- Can failover multiple times without operator intervention
- Takeover of queue manager's IP address as part of the failover

Limitations of HA clusters include:
- Additional product purchase and skills are required
- Disks which can be switched between the nodes of the cluster are required
- Configuration of HA clusters is relatively complex
- Failover is rather slow historically, but recent HA cluster products are improving this
- Unnecessary failovers can occur if there are shortcomings in the scripts that are used to monitor resources such as queue managers

Relationship of HA clusters to queue manager clusters

Queue manager clusters provide load balancing of messages across available instances of queue manager cluster queues. This offers higher availability than a single queue manager because, following a failure of a queue manager, messaging applications can still send messages to, and access, surviving instances of a queue manager cluster queue. However, although queue manager clusters automatically route new messages to the available queue managers in a cluster, messages currently queued on an unavailable queue manager are not available until that queue manager is restarted. For this reason, queue manager clusters alone do not provide high availability of all message data or provide automatic detection of queue manager failure and automatic triggering of queue manager restart or failover. High Availability (HA) clusters provide these features. The two types of cluster can be used together to good effect. For an introduction to queue manager clusters, see Designing clusters.

HA clusters on UNIX and Linux

UNIX Linux

You can use IBM MQ with a high availability (HA) cluster on UNIX and Linux platforms: for example, PowerHA for AIX (formerly HACMP), Veritas Cluster Server, HP Serviceguard, or a Red Hat Enterprise Linux cluster with Red Hat Cluster Suite.

Before IBM WebSphere MQ Version 7.0.1, SupportPac MC91 was provided to assist in configuring HA clusters. IBM WebSphere MQ Version 7.0.1 provided a greater degree of control than previous versions over where queue managers store their data. This makes it easier to configure queue managers in an HA cluster. Most of the scripts provided with SupportPac MC91 are no longer required, and the SupportPac is withdrawn.

This section introduces “HA cluster configurations” on page 1210, the relationship of HA clusters to queue manager clusters, “IBM MQ clients” on page 1210, and “IBM MQ operating in an HA cluster” on page 1211, and guides you through the steps and provides example scripts that you can adapt to configure queue managers with an HA cluster.

Refer to the HA cluster documentation particular to your environment for assistance with the configuration steps described in this section.

HA cluster configurations

In this section the term *node* is used to refer to the entity that is running an operating system and the HA software; "computer", "system" or "machine" or "partition" or "blade" might be considered synonyms in this usage. You can use IBM MQ to help set up either standby or takeover configurations, including mutual takeover where all cluster nodes are running IBM MQ workload.

A *standby* configuration is the most basic HA cluster configuration in which one node performs work while the other node acts only as standby. The standby node does not perform work and is referred to as idle; this configuration is sometimes called *cold standby*. Such a configuration requires a high degree of hardware redundancy. To economize on hardware, it is possible to extend this configuration to have multiple worker nodes with a single standby node. The point of this is that the standby node can take over the work of any other worker node. This configuration is still referred to as a standby configuration and sometimes as an "N+1" configuration.

A *takeover* configuration is a more advanced configuration in which all nodes perform some work and critical work can be taken over in the event of a node failure.

A *one-sided takeover* configuration is one in which a standby node performs some additional, noncritical and unmovable work. This configuration is similar to a standby configuration but with (noncritical) work being performed by the standby node.

A *mutual takeover* configuration is one in which all nodes are performing highly available (movable) work. This type of HA cluster configuration is also sometimes referred to as "Active/Active" to indicate that all nodes are actively processing critical workload.

With the extended standby configuration or either of the takeover configurations it is important to consider the peak load that might be placed on a node that can take over the work of other nodes. Such a node must possess sufficient capacity to maintain an acceptable level of performance.

Relationship of HA clusters to queue manager clusters

Queue manager clusters reduce administration and provide load balancing of messages across instances of queue manager cluster queues. They also offer higher availability than a single queue manager because, following a failure of a queue manager, messaging applications can still access surviving instances of a queue manager cluster queue. However, queue manager clusters alone do not provide automatic detection of queue manager failure and automatic triggering of queue manager restart or failover. HA clusters provide these features. The two types of cluster can be used together to good effect.

IBM MQ clients

IBM MQ clients that are communicating with a queue manager that might be subject to a restart or takeover must be written to tolerate a broken connection and must repeatedly attempt to reconnect. IBM WebSphere MQ Version 7 introduced features in the processing of the Client Channel Definition Table (CCDT) that assist with connection availability and workload balancing; however these are not directly relevant when working with a failover system.

Transactional functionality allows an IBM MQ MQI client to participate in two-phase transactions, as long as the client is connected to the same queue manager. Transactional functionality cannot use techniques, such as an IP load balancer, to select from a list of queue managers. When you use an HA product, a queue manager maintains its identity (name and address) whichever node it is running on, so transactional functionality can be used with queue managers that are under HA control.

IBM MQ operating in an HA cluster

All HA clusters have the concept of a unit of failover. This is a set of definitions that contains all the resources that make up the highly available service. The unit of failover includes the service itself and all other resources upon which it depends.

HA solutions use different terms for a unit of failover:
- On PowerHA for AIX the unit of failover is called a *resource group*.
- On Veritas Cluster Server it is known as a *service group*.
- On Serviceguard it is called a *package*.

This topic uses the term *resource group* to mean a unit of failover.

The smallest unit of failover for IBM MQ is a queue manager. Typically, the resource group containing the queue manager also contains shared disks in a volume group or disk group that is reserved exclusively for use by the resource group, and the IP address that is used to connect to the queue manager. It is also possible to include other IBM MQ resources, such as a listener or a trigger monitor in the same resource group, either as separate resources, or under the control of the queue manager itself.

A queue manager that is to be used in an HA cluster must have its data and logs on disks that are shared between the nodes in the cluster. The HA cluster ensures that only one node in the cluster at a time can write to the disks. The HA cluster can use a monitor script to monitor the state of the queue manager.

It is possible to use a single shared disk for both the data and logs that are related to the queue manager. However, it is normal practice to use separate shared file systems so that they can be independently sized and tuned.

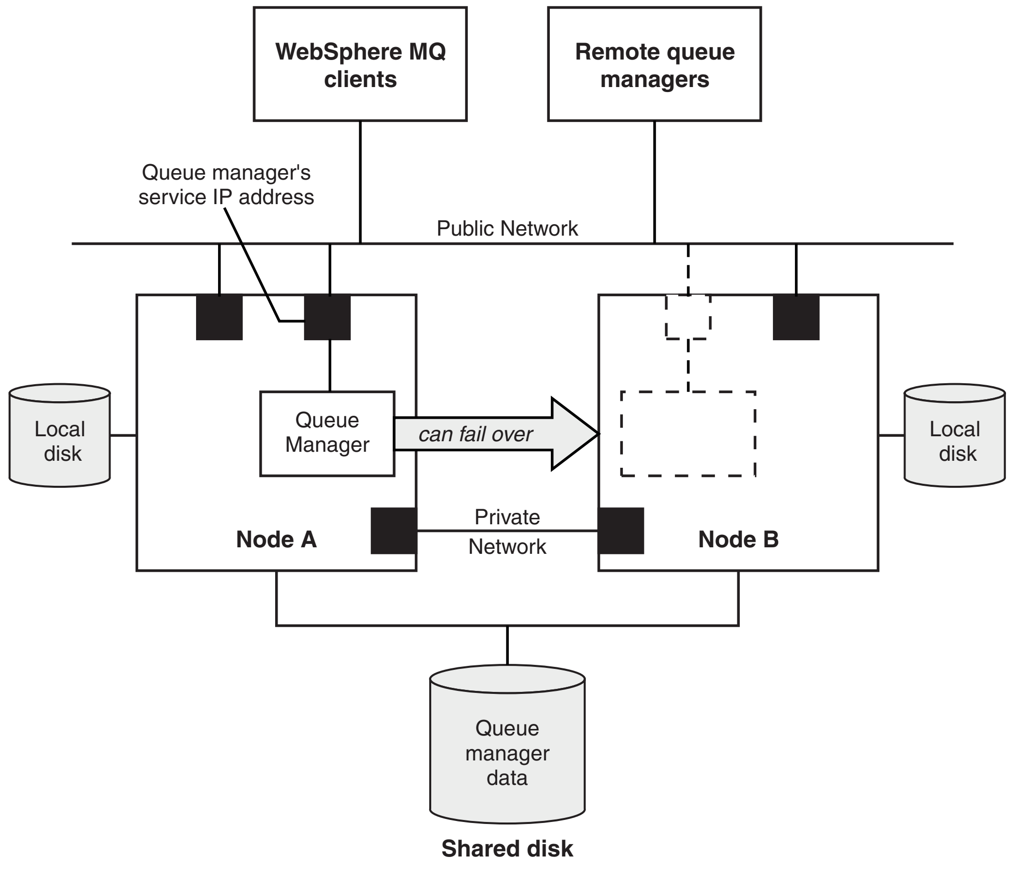


Figure 149. HA cluster

Figure 1 illustrates a HA cluster with two nodes. The HA cluster is managing the availability of a queue manager which has been defined in a resource group. This is an active/passive or cold standby configuration, because only one node, node A, is currently running a queue manager. The queue manager was created with its data and log files on a shared disk. The queue manager has a service IP address which is also managed by the HA cluster. The queue manager depends on the shared disk and its service IP address. When the HA cluster fails the queue manager over from node A to node B, it first moves the queue manager's dependent resources onto node B and then starts the queue manager.

If the HA cluster contains more than one queue manager, your HA cluster configuration might result in two or more queue managers running on the same node after a failover. Each queue manager in the HA cluster must be assigned its own port number, which it uses on whichever cluster node it happens to be active at any particular time.

Generally, the HA cluster runs as the root user. IBM MQ runs as the mqm user. Administration of IBM MQ is granted to members of the mqm group. Ensure that the mqm user and group both exist on all HA cluster nodes. The user ID and group ID must be consistent across the cluster. Administration of IBM MQ by the root user is not allowed; scripts that start, stop, or monitor scripts must switch to the mqm user.

Note: IBM MQ must be installed correctly on all nodes; you cannot share the product executable files.

Configuring shared disks on UNIX and Linux: UNIX Linux

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.

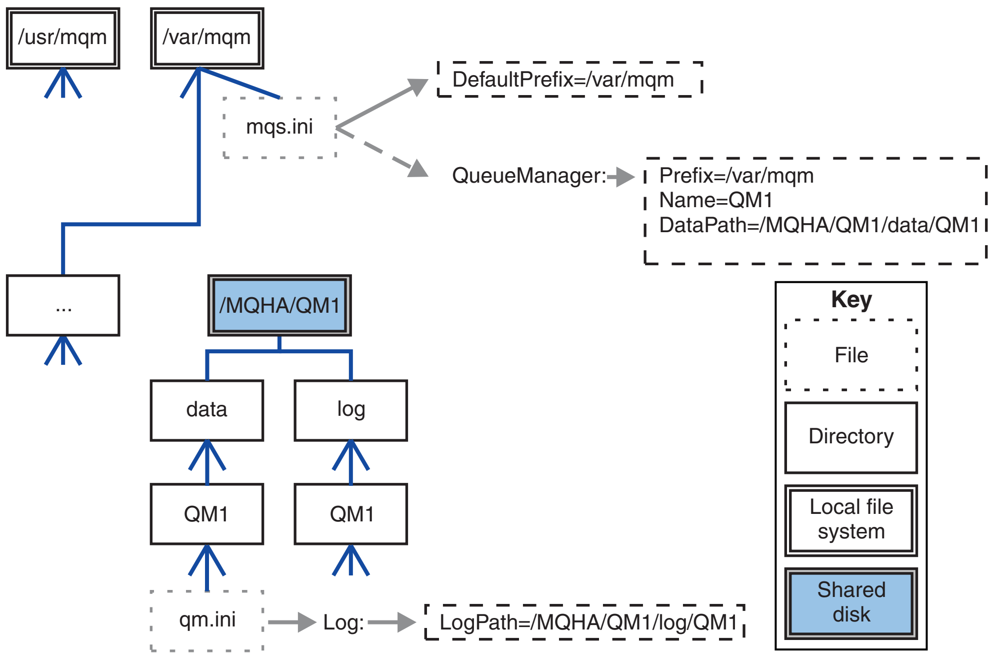


Figure 150. Shared named `data` and `log` directories

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 UNIX and Linux: UNIX 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 UNIX and Linux.” 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 UNIX and Linux.”

Adding queue manager configuration to other HA cluster nodes on UNIX and Linux: UNIX 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 UNIX and Linux.” 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 UNIX and Linux.”

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 UNIX and Linux" on page 1214.
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.

Starting an HA cluster queue manager on UNIX and Linux: UNIX 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 amqrrmfa amqrmppa \
              amqzfuma amqzmuf0 amqzmur0 amqzmgr0
 do
  ps -ef | tr "\t" " " | grep $process | grep -v grep | \
    egrep "$srchstr" | awk '{print $2}'| \
       xargs kill -9 > /dev/null 2>&1
done

# It is now safe to start the queue manager.
# The strmqm command does not use the -x flag.
strmqm ${QM}
```

You can modify the script to start other related programs.

Stopping an HA cluster queue manager on UNIX and Linux: UNIX 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 without making assumptions about the current state of the queue manager. The script must be run by the mqm user; it might therefore be necessary to wrap this script in a shell script to switch the user from the HA cluster user to mqm (an example shell script is provided in "Starting an HA cluster queue manager on UNIX and Linux" on page 1215):

```
#!/bin/ksh
#
# The script ends the QM by using two phases, initially trying an immediate
# end with a time-out and escalating to a forced stop of remaining
# processes.
#
# The script must be run by the mqm user.
#
# There are two arguments: the queue manager name and a timeout value.
QM=$1
TIMEOUT=$2

if [ -z "$QM" ]
then
  echo "ERROR! No queue manager name supplied"
  exit 1
fi

if [ -z "$TIMEOUT" ]
then
  echo "ERROR! No timeout specified"
  exit 1
fi

for severity in immediate brutal
do
  # End the queue manager in the background to avoid
  # it blocking indefinitely. Run the TIMEOUT timer
  # at the same time to interrupt the attempt, and try a
  # more forceful version. If the brutal version fails,
  # nothing more can be done here.

  echo "Attempting ${severity} end of queue manager '${QM}'"
  case $severity in

  immediate)
    # Minimum severity of endmqm is immediate which severs connections.
    # HA cluster should not be delayed by clients
    endmqm -i ${QM} &
    ;;
  brutal)
    # This is a forced means of stopping queue manager processes.

    srchstr="( |-m)$QM *.*$"
    for process in amqzmuc0 amqzxma0 amqfcxba amqfqpub amqpcsea amqzlaa0 \
               amqzlsa0 runmqchi runmqlsr amqcrsta amqrrmfa amqrmppa \
               amqzfuma amqzmuf0 amqzmur0 amqzmgr0
    do
      ps -ef | tr "\t" " " | grep $process | grep -v grep | \
         egrep "$srchstr" | awk '{print $2}'| \
            xargs kill -9 > /dev/null 2>&1
    done

  esac

  TIMED_OUT=yes
  SECONDS=0
  while (( $SECONDS < ${TIMEOUT} ))
  do
   TIMED_OUT=yes
   i=0
   while [ $i -lt 5 ]
   do
     # Check for execution controller termination
     srchstr="( |-m)$QM *.*$"
     cnt=`ps -ef | tr "\t" " " | grep amqzxma0 | grep -v grep | \
       egrep "$srchstr" | awk '{print $2}' | wc -l `
     i=`expr $i + 1`
     sleep 1
     if [ $cnt -eq 0 ]
     then
       TIMED_OUT=no
       break
     fi
   done

   if [ ${TIMED_OUT} = "no" ]
   then
     break
   fi

   echo "Waiting for ${severity} end of queue manager '${QM}'"
   sleep 1
  done # timeout loop

  if [ ${TIMED_OUT} = "yes" ]
  then
    continue        # to next level of urgency
  else
    break           # queue manager is ended, job is done
  fi

done # next phase
```

Monitoring an HA cluster queue manager on UNIX and Linux: UNIX 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.

From IBM WebSphere MQ Version 7.1, it is possible to have multiple installations of IBM MQ coexisting on a system. For more information about multiple installations, see Multiple installations. If you intend to use the monitoring script across multiple installations, including installations at Version 7.1, or higher, you might need to perform some additional steps. If you have a primary installation, or you are using the script with versions earlier than Version 7.1, you do not need to specify the *MQ_INSTALLATION_PATH* to use the script. Otherwise, the following steps ensure that the *MQ_INSTALLATION_PATH* is identified correctly:

1. Use the **crtmqenv** command from a Version 7.1 installation to identify the correct *MQ_INSTALLATION_PATH* for a queue manager:

   ```
   crtmqenv -m qmname
   ```

   This command returns the correct *MQ_INSTALLATION_PATH* value for the queue manager specified by *qmname*.
2. Run the monitoring script with the appropriate *qmname* and *MQ_INSTALLATION_PATH* parameters.

Note: PowerHA for AIX does not provide a way of supplying a parameter to the monitoring program for the queue manager. You must create a separate monitoring program for each queue manager, that encapsulates the queue manager name. Here is an example of a script used on AIX to encapsulate the queue manager name:

```
#!/bin/ksh
su mqm -c name_of_monitoring_script qmname  MQ_INSTALLATION_PATH
```

where *MQ_INSTALLATION_PATH* is an optional parameter that specifies the path to the installation of IBM MQ that the queue manager *qmname* is associated with.

The following script is not robust to the possibility that **runmqsc** hangs. Typically, HA clusters treat a hanging monitoring script as a failure and are themselves robust to this possibility.

The script does, however, tolerate the queue manager being in the starting state. This is because it is common for the HA cluster to start monitoring the queue manager as soon as it has started it. Some HA clusters distinguish between a starting phase and a running phase for resources, but it is necessary to configure the duration of the starting phase. Because the time taken to start a queue manager depends on the amount of work that it has to do, it is hard to choose a maximum time that starting a queue manager takes. If you choose a value that is too low, the HA cluster incorrectly assumes that the queue manager failed when it has not completed starting. This could result in an endless sequence of failovers.

This script must be run by the mqm user; it might therefore be necessary to wrap this script in a shell script to switch the user from the HA cluster user to mqm (an example shell script is provided in "Starting an HA cluster queue manager on UNIX and Linux" on page 1215):

```
#!/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 UNIX and Linux: UNIX 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 UNIX and Linux: UNIX 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 **dltmqm** command.

   Important: Be aware that deleting the queue manager by using the **dltmqm** 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 the Microsoft Cluster Service (MSCS)

Introducing and setting up MSCS to support failover of virtual servers.

This information applies to IBM MQ for Windows only.

The Microsoft Cluster Service (MSCS) enables you to connect servers into a *cluster*, giving higher availability of data and applications, and making it easier to manage the system. MSCS can automatically detect and recover from server or application failures.

MSCS supports *failover* of *virtual servers*, which correspond to applications, Web sites, print queues, or file shares (including, for example, their disk spindles, files, and IP addresses).

Failover is the process by which MSCS detects a failure in an application on one computer in the cluster, and shuts down the disrupted application in an orderly manner, transfers its state data to the other computer, and reinitiates the application there.

This section introduces MSCS clusters and describes setting up MSCS support in the following sections:
- "Introducing MSCS clusters"
- "Setting up IBM MQ for MSCS clustering" on page 1223

Then tells you how to configure IBM MQ for MSCS clustering, in the following sections:
- "Creating a queue manager for use with MSCS" on page 1225
- "Moving a queue manager to MSCS storage" on page 1226
- "Putting a queue manager under MSCS control" on page 1227
- "Removing a queue manager from MSCS control" on page 1233

And then gives some useful hints on using MSCS with IBM MQ, and details the IBM MQ MSCS support utility programs, in the following sections:
- "Hints and tips on using MSCS" on page 1234
- "Support for MSCS utility programs" on page 1237

Introducing MSCS clusters:

MSCS clusters are groups of two or more computers, connected together and configured in such a way that, if one fails, MSCS performs a *failover*, transferring the state data of applications from the failing computer to another computer in the cluster and re-initiating their operation there.

"High availability configurations" on page 1208 contains a comparison between MSCS clusters, multi-instance queue managers, and IBM MQ clusters.

In this section and its subordinate topics, the term *cluster*, when used by itself, **always** means an MSCS cluster. This is distinct from an IBM MQ cluster described elsewhere in this guide.

A two-machine cluster comprises two computers (for example, A and B) that are jointly connected to a network for client access using a *virtual IP address*. They might also be connected to each other by one or more private networks. A and B share at least one disk for the server applications on each to use. There is also another shared disk, which must be a redundant array of independent disks (*RAID*) Level 1, for the exclusive use of MSCS; this is known as the *quorum* disk. MSCS monitors both computers to check that the hardware and software are running correctly.

In a simple setup such as this, both computers have all the applications installed on them, but only computer A runs with live applications; computer B is just running and waiting. If computer A encounters any one of a range of problems, MSCS shuts down the disrupted application in an orderly manner, transfers its state data to the other computer, and re-initiates the application there. This is known as a *failover*. Applications can be made *cluster-aware* so that they interact fully with MSCS and failover gracefully.

A typical setup for a two-computer cluster is as shown in Figure 151.
Each computer can access the shared disk, but only one at a time, under the control of MSCS. In the

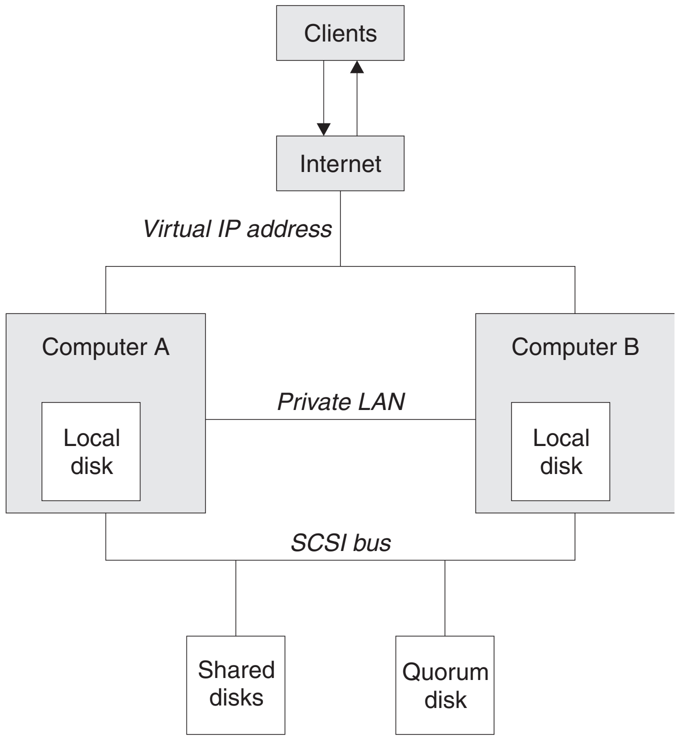


Figure 151. Two-computer MSCS cluster

event of a failover, MSCS switches the access to the other computer. The shared disk itself is usually a RAID, but need not be.

Each computer is connected to the external network for client access, and each has an IP address. However an external client, communicating with this cluster, is aware of only one *virtual IP address*, and MSCS routes the IP traffic within the cluster appropriately.

MSCS also performs its own communications between the two computers, either over one or more private connections or over the public network, for example to monitor their states using the heartbeat, and to synchronize their databases.

Setting up IBM MQ for MSCS clustering: ▶ Windows

You configure IBM MQ for clustering by making the queue manager the unit of failover to MSCS. You define a queue manager as a resource to MSCS, which can then monitor it, and transfer it to another computer in the cluster if there is a problem.

To set your system up for this, you start by installing IBM MQ on each computer in the cluster.

As the queue manager is associated with the IBM MQ installation name, the IBM MQ installation name on all the computers in the cluster should be the same. See Installing and uninstalling.

The queue managers themselves need to exist only on the computer on which you create them. In the event of a failover, the MSCS initiates the queue managers on the other computer. The queue managers, however, must have their log and data files on a cluster shared disk, and not on a local drive. If you have a queue manager already installed on a local drive, you can migrate it using a tool provided with IBM MQ; see “Moving a queue manager to MSCS storage” on page 1226. If you want to create new queue managers for use with MSCS, see “Creating a queue manager for use with MSCS” on page 1225.

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 1227.

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 1233.

Setup symmetry and MSCS: ▶ Windows

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.”

MSCS security: ▶ Windows

For successful MSCS security, follow these guidelines.

The guidelines are as follows:
- Make sure you that you have identical software installations on each computer in the cluster.
- Create a common namespace (security environment) across the cluster.
- Make the nodes of the MSCS cluster members of a domain, within which the user account that is the *cluster owner* is a domain account.
- Make the other user accounts on the cluster also domain accounts, so that they are available on both nodes. This is automatically the case if you already have a domain, and the accounts relevant to IBM MQ are domain accounts. If you do not currently have a domain, consider setting up a *mini-domain* to cater for the cluster nodes and relevant accounts. Your aim is to make your cluster of two computers look like a single computing resource.

  Remember that an account that is local to one computer does not exist on the other one. Even if you create an account with the same name on the other computer, its security identifier (SID) is different, so, when your application is moved to the other node, the permissions do not exist on that node.

During a failover or move, IBM MQ MSCS support ensures that all files that contain queue manager objects have equivalent permissions on the destination node. Explicitly, the code checks that the

Administrators and mqm groups, and the SYSTEM account, have full control, and that if `Everyone` had read access on the old node, that permission is added on the destination node.

You can use a domain account to run your IBM MQ Service. Make sure that it exists in the local mqm group on each computer in the cluster.

Using multiple queue managers with MSCS: Windows

If you are running more than one queue manager on a computer, you can choose one of these setups.

The setups are as follows:
- All the queue managers in a single group. In this configuration, if a problem occurs with any queue manager, all the queue managers in the group failover to the other computer as a group.
- A single queue manager in each group. In this configuration, if a problem occurs with the queue manager, it alone fails over to the other computer without affecting the other queue managers.
- A mixture of the first two setups.

Cluster modes and MSCS: Windows

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: Windows

This procedure ensures that a new queue manager is created in such a way that it is suitable for preparing and placing under MSCS control.

You start by creating the queue manager with all its resources on a local drive, and then migrate the log files and data files to a shared disk. (You can reverse this operation.) Do **not** attempt to create a queue manager with its resources on a shared drive.

You can create a queue manager for use with MSCS in two ways, either from a command prompt, or in the IBM MQ Explorer. The advantage of using a command prompt is that the queue manager is created *stopped* and set to *manual startup*, which is ready for MSCS. (The IBM MQ Explorer automatically starts a new queue manager and sets it to automatic startup after creation. You have to change this.)

Creating a queue manager from a command prompt

Follow these steps to create a queue manager from a command prompt, for use with MSCS:
1. Ensure that you have the environment variable `MQSPREFIX` set to refer to a local drive, for example `C:\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 1226.

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 1226.

Moving a queue manager to MSCS storage: Windows

This procedure configures an existing queue manager to make it suitable for putting under MSCS control.

To achieve this, you move the log files and data files to shared disks to make them available to the other computer in the event of a failure. For example, the existing queue manager might have paths such as `C:\WebSphere MQ\log\`*`QMname`* and `C:\WebSphere MQ\qmgrs\`*`QMname`*.

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.

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 1227.
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 1235 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."

Putting a queue manager under MSCS control: Windows

The tasks involved in placing a queue manager under MSCS control, including prerequisite tasks.

Before you put a queue manager under MSCS control

Before you put a queue manager under MSCS control, perform the following tasks:
1. Ensure that IBM MQ and its MSCS Support are installed on both machines in the cluster and that the software on each computer is identical, as described in "Setting up IBM MQ for MSCS clustering" on page 1223.
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 1237 for additional information.
3. If you have not yet created the queue manager, see "Creating a queue manager for use with MSCS" on page 1225.
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 1226.
5. Stop the queue manager, if it is running, using either a command prompt or the IBM MQ Explorer.
6. Test MSCS operation of the shared drives before going on to either of the following Windows procedures in this topic.

Windows Server 2012

To place a queue manager under MSCS control on Windows Server 2012, use the following procedure:
1. Log in to the cluster node computer hosting the queue manager, or log in to a remote workstation as a user with cluster administration permissions, and connect to the cluster node hosting the queue manager.
2. Start the Failover Cluster Management tool.
3. Right-click **Failover Cluster Management > Connect Cluster ...** to open a connection to the cluster.
4. In contrast to the group scheme used in the MSCS Cluster Administrator on previous versions of Windows, the Failover Cluster Management tool uses the concept of services and applications. A configured service or application contains all the resources necessary for one application to be clustered. You can configure a queue manager under MSCS as follows:
   a. Right-click on the cluster and select **Configure Role** to start the configuration wizard.
   b. Select **Other Server** on the "Select Service or Application" panel.
   c. Select an appropriate IP address as a client access point.

      This address should be an unused IP address to be used by clients and other queue managers to connect to the *virtual* queue manager. This IP address is not the normal (static) address of either node; it is an additional address that *floats* between them. Although MSCS handles the routing of this address, it does **not** verify that the address can be reached.
   d. Assign a storage device for exclusive use by the queue manager. This device needs to be created as a resource instance before it can be assigned.

      You can use one drive to store both the logs and queue files, or you can split them up across drives. In either case, if each queue manager has its own shared disk, ensure that all drives used by this queue manager are exclusive to this queue manager, that is, that nothing else relies on the drives. Also ensure that you create a resource instance for every drive that the queue manager uses.

The resource type for a drive depends on the SCSI support you are using; refer to your SCSI adapter instructions. There might already be groups and resources for each of the shared drives. If so, you do not need to create the resource instance for each drive. Move it from its current group to the one created for the queue manager.

For each drive resource, set possible owners to both nodes. Set dependent resources to none.

   e. Select the **MQSeries MSCS** resource on the "Select Resource Type" panel.
   f. Complete the remaining steps in the wizard.
5. Before bringing the resource online, the MQSeries MSCS resource needs additional configuration:
   a. Select the newly defined service which contains a resource called 'New MQSeries MSCS'.
   b. Right-click **Properties** on the MQ resource.
   c. Configure the resource:
      - `Name` ; choose a name that makes it easy to identify which queue manager it is for.
      - `Run in a separate Resource Monitor` ; for better isolation
      - `Possible owners` ; set both nodes
      - `Dependencies` ; add the drive and IP address for this queue manager.

        Warning: Failure to add these dependencies means that IBM MQ attempts to write the queue manager status to the wrong cluster disk during failovers. Because many processes might be attempting to write to this disk simultaneously, some IBM MQ processes could be blocked from running.
      - `Parameters` ; as follows:
        - `QueueManagerName` (required); the name of the queue manager that this resource is to control. This queue manager must exist on the local computer.
        - `PostOnlineCommand` (optional); you can specify a program to run whenever the queue manager resource changes its state from offline to online. For more details see "PostOnlineCommand and PreOfflineCommand in MSCS" on page 1236.
        - `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 1236.

          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 1232.
   d. Optionally, set a preferred node (but note the comments in "Using preferred nodes in MSCS" on page 1237)
   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 1226.) You can also right-click and select `Initiate Failure` ; the action (local restart or failover) depends on the current state and the configuration settings.

Windows Server 2008

To place a queue manager under MSCS control on Windows Server 2008, use the following procedure:

1. Log in to the cluster node computer hosting the queue manager, or log in to a remote workstation as a user with cluster administration permissions, and connect to the cluster node hosting the queue manager.
2. Start the Failover Cluster Management tool.
3. Right-click **Failover Cluster Management > Manage a Cluster ...** to open a connection to the cluster.
4. In contrast to the group scheme used in the MSCS Cluster Administrator on previous versions of Windows, the Failover Cluster Management tool uses the concept of services and applications. A configured service or application contains all the resources necessary for one application to be clustered. You can configure a queue manager under MSCS as follows:
   a. Right-click **Services and Applications > Configure a Service or Application ...** to start the configuration wizard.
   b. Select **Other Server** on the Select Service or Application panel.
   c. Select an appropriate IP address as a client access point.

      This address should be an unused IP address to be used by clients and other queue managers to connect to the *virtual* queue manager. This IP address is not the normal (static) address of either node; it is an additional address that *floats* between them. Although MSCS handles the routing of this address, it does **not** verify that the address can be reached.
   d. Assign a storage device for exclusive use by the queue manager. This device needs to be created as a resource instance before it can be assigned.

      You can use one drive to store both the logs and queue files, or you can split them up across drives. In either case, if each queue manager has its own shared disk, ensure that all drives used by this queue manager are exclusive to this queue manager, that is, that nothing else relies on the drives. Also ensure that you create a resource instance for every drive that the queue manager uses.

      The resource type for a drive depends on the SCSI support you are using; refer to your SCSI adapter instructions. There might already be groups and resources for each of the shared drives. If so, you do not need to create the resource instance for each drive. Move it from its current group to the one created for the queue manager.

      For each drive resource, set possible owners to both nodes. Set dependent resources to none.
   e. Select the **MQSeries MSCS** resource on the Select Resource Type panel.
   f. Complete the remaining steps in the wizard.
5. Before bringing the resource online, the MQSeries MSCS resource needs additional configuration:
   a. Select the newly defined service which contains a resource called 'New MQSeries MSCS'.
   b. Right-click **Properties** on the MQ resource.
   c. Configure the resource:

- `Name` ; choose a name that makes it easy to identify which queue manager it is for.
- `Run in a separate Resource Monitor` ; for better isolation
- `Possible owners` ; set both nodes
- `Dependencies` ; add the drive and IP address for this queue manager.

  Warning: Failure to add these dependencies means that IBM MQ attempts to write the queue manager status to the wrong cluster disk during failovers. Because many processes might be attempting to write to this disk simultaneously, some IBM MQ processes could be blocked from running.
- `Parameters` ; as follows:
  - `QueueManagerName` (required); the name of the queue manager that this resource is to control. This queue manager must exist on the local computer.
  - `PostOnlineCommand` (optional); you can specify a program to run whenever the queue manager resource changes its state from offline to online. For more details see "PostOnlineCommand and PreOfflineCommand in MSCS" on page 1236.
  - `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 1236.

    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 1232.

d. Optionally, set a preferred node (but note the comments in "Using preferred nodes in MSCS" on page 1237)

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 1226.) You can also right-click and select `Initiate Failure` ; the action (local restart or failover) depends on the current state and the configuration settings.

Windows 2003

To place a queue manager under MSCS control on Windows 2003, use the following procedure:

1. Log in to the cluster node computer hosting the queue manager, or log in to a remote workstation as a user with cluster administration permissions, and connect to the cluster node hosting the queue manager.
2. Start the MSCS Cluster Administrator.
3. Open a connection to the cluster.
4. Create an MSCS group to be used to contain the resources for the queue manager. Name the group in such a way that it is obvious which queue manager it relates to. Each group can contain multiple queue managers, as described in "Using multiple queue managers with MSCS" on page 1224.

   Use the group for all the remaining steps.
5. Create a resource instance for each of the SCSI logical drives that the queue manager uses.

   You can use one drive to store both the logs and queue files, or you can split them up across drives. In either case, if each queue manager has its own shared disk, ensure that all drives used by this queue manager are exclusive to this queue manager, that is, that nothing else relies on the drives. Also ensure that you create a resource instance for every drive that the queue manager uses.

   The resource type for a drive depends on the SCSI support you are using; refer to your SCSI adapter instructions. There might already be groups and resources for each of the shared drives. If so, you do not need to create the resource instance for each drive. Move it from its current group to the one created for the queue manager.

   For each drive resource, set possible owners to both nodes. Set dependent resources to none.
6. Create a resource instance for the IP address.

   Create an IP address resource (resource type *IP address*). This address should be an unused IP address to be used by clients and other queue managers to connect to the *virtual* queue manager. This IP address is not the normal (static) address of either node; it is an additional address that *floats* between them. Although MSCS handles the routing of this address, it does **not** verify that the address can be reached.
7. Create a resource instance for the queue manager.

   Create a resource of type *IBM MQ MSCS*. The wizard prompts you for various items, including the following:
   - `Name` ; choose a name that makes it easy to identify which queue manager it is for.
   - `Add to group` ; use the group that you created
   - `Run in a separate Resource Monitor` ; for better isolation
   - `Possible owners` ; set both nodes
   - `Dependencies` ; add the drive and IP address for this queue manager.

     Warning: Failure to add these dependencies means that IBM MQ attempts to write the queue manager status to the wrong cluster disk during failovers. Because many processes might be attempting to write to this disk simultaneously, some IBM MQ processes could be blocked from running.
   - `Parameters` ; as follows:
     - `QueueManagerName` (required); the name of the queue manager that this resource is to control. This queue manager must exist on the local computer.
     - `PostOnlineCommand` (optional); you can specify a program to run whenever the queue manager resource changes its state from offline to online. For more details see "PostOnlineCommand and PreOfflineCommand in MSCS" on page 1236.
     - `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 1236.

Note: The *looksAlive* poll interval is set to default value of 5000 ms. The *isAlive* poll interval is set to default value of 30000 ms. These defaults can only be modified after the resource definition has been completed. For further details see "looksAlive and isAlive polling on MSCS."

8. Optionally, set a preferred node (but note the comments in "Using preferred nodes in MSCS" on page 1237)
9. The *Failover Policy* (as defined in the properties for the group) is set by default to sensible values, but you can tune the thresholds and periods that control *Resource Failover* and *Group Failover* to match the loads placed on the queue manager.
10. Test the queue manager by bringing it online in the MSCS Cluster Administrator and subjecting it to a test workload. If you are experimenting with a test queue manager, use the IBM MQ Explorer. For example:
    a. Right-click the `Queues` tree node, then select **New** > **Local Queue...**, and give the queue a name.
    b. Click **Finish**. The queue is created, and displayed in the content view.
    c. Right-click the queue, then select **Put Test Message...**. The Put Test Message panel is displayed.
    d. Type some message text, then click **Put Test Message**, and close the panel.
    e. Right-click the queue, then select **Browse Messages...**. The Message Browser panel is displayed.
    f. Ensure that your message is on the queue, then click **Close**. The Message Browser panel closes.
    g. Right-click the queue, then select **Clear Messages...**. The messages on the queue are cleared.
    h. Right-click the queue, then select **Delete...**. A confirmation panel is displayed, click **OK**. The queue is deleted.
11. Test that the queue manager can be taken offline and back online using the MSCS Cluster Administrator.
12. Simulate a failover.

    In the MSCS Cluster Administrator, right-click the group containing the queue manager and select `Move Group`. This can take some minutes to do. (If at other times you want to move a queue manager to another node quickly, follow the procedure in "Moving a queue manager to MSCS storage" on page 1226.) 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: Windows

looksAlive and *isAlive* are intervals at which MSCS calls back into the resource types supplied library code and requests that the resource performs checks to determine the working status of itself. This ultimately determines if MSCS attempts to fail over the resource.

On every occasion that the *looksAlive* interval elapses (default 5000 ms), the queue manager resource is called to perform its own check to determine if its status is satisfactory.

On every occasion that the *isAlive* interval elapses (default 30000 ms), another call is made to the queue manager resource for it to perform another check to determine if the resource is functioning correctly. This enables two levels of resource type checking.

1. A *looksAlive* status check to establish if the resource appears to be functioning.
2. A more significant *isAlive* check that determines if the queue manager resource is active.

If the queue manager resource is determined not to be active, MSCS, based on other advanced MSCS options, triggers a fail over for the resource and associated dependant resources to another node in the cluster. For further information, see MSCS documentation.

Removing a queue manager from MSCS control: ▶ Windows

You can remove queue managers from MSCS control, and return them to manual administration.

You do not need to remove queue managers from MSCS control for maintenance operations. You can do that by taking a queue manager offline temporarily, using the MSCS Cluster Administrator. Removing a queue manager from MSCS control is a more permanent change; only do it if you decide that you no longer want MSCS to have any further control of the queue manager.

If the queue manager is being removed uses TSL 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"
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."
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 1235 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 1226).

Hints and tips on using MSCS:

This section contains some general information to help you use IBM MQ support for MSCS effectively.

This section contains some general information to help you use IBM MQ support for MSCS effectively.

How long does it take to fail a queue manager over from one machine to the other? This depends heavily on the amount of workload on the queue manager and on the mix of traffic, for example, how much of it is persistent, within sync point, and how much committed before the failure. IBM tests have given failover and failback times of about a minute. This was on a very lightly loaded queue manager and actual times will vary considerably depending on load.

Verifying that MSCS is working:

Follow these steps to ensure that you have a running MSCS cluster.

The task descriptions starting with "Creating a queue manager for use with MSCS" on page 1225 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 1226 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 1226, 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 1237 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 151 on page 1222). 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 1225
- “Moving a queue manager to MSCS storage” on page 1226
- “Putting a queue manager under MSCS control” on page 1227

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 1226).

It is advisable before you move, to shut the queue manager cleanly and take a full backup of the data files and log files.

Multiple queue managers

IBM MQ MSCS support allows you to run multiple queue managers on each machine and to place individual queue managers under MSCS control.

Always use MSCS to manage clusters: Windows

Do not try to perform start and stop operations directly on any queue manager under the control of MSCS, using either the control commands or the IBM MQ Explorer. Instead, use MSCS Cluster Administrator to bring the queue manager online or take it offline.

Using the MSCS Cluster Administrator is partly to prevent possible confusion caused by MSCS reporting that the queue manager is offline, when in fact you have started it outside the control of MSCS. More seriously, stopping a queue manager without using MSCS is detected by MSCS as a failure, initiating failover to the other node.

Working in Active/Active mode in MSCS: Windows

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: Windows

Use these commands to integrate IBM MQ MSCS support with other systems. You can use them to issue IBM MQ commands, wih some restrictions.

Specify these commands in the Parameters to a resource of type `IBM MQ MSCS`. You can use them to integrate IBM MQ MSCS support with other systems or procedures. For example, you could specify the name of a program that sends a mail message, activates a pager, or generates some other form of alert to be captured by another monitoring system.

PostOnlineCommand is invoked when the resource changes from offline to online; PreOfflineCommand is invoked for a change from online to offline. When invoked these commands are run, by default, from the Windows system directory. Because IBM MQ uses a 32-bit resource monitor process, on Windows 64-bit systems, this is the `\Windows\SysWOW64` directory rather than the `\Windows\system32` directory. For more information, see the Microsoft documentation about file redirection in a Windows x64 environment. Both commands run under the user account used to run the MSCS Cluster Service; and are invoked asynchronously; IBM MQ MSCS support does not wait for them to complete before continuing. This eliminates any risk that they might block or delay further cluster operations.

You can also use these commands to issue IBM MQ commands, for example to restart Requester channels. However, the commands are run at the point in time when the queue manager's state changes so they are not intended to perform long-running functions and must not make assumptions about the current state of the queue manager; it is quite possible that, immediately after the queue manager was brought online, an administrator issued an offline command.

If you want to run programs that depend on the state of the queue manager, consider creating instances of the `MSCS Generic Application` resource type, placing them in the same MSCS group as the queue manager resource, and making them dependent on the queue manager resource.

Using preferred nodes in MSCS: Windows

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: Windows

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.

Support for MSCS utility programs: Windows

A list of the IBM MQ support for MSCS utility programs that you can run at a command prompt.

IBM MQ support for MSCS includes the following utility programs:

Register/unregister the resource type

`haregtyp.exe`

After you *unregister* the IBM MQ MSCS resource type you can no longer create any resources of that type. MSCS does not let you unregister a resource type if you still have instances of that type within the cluster:

1. Using the MSCS Cluster Administrator, stop any queue managers that are running under MSCS control, by taking them offline as described in "Taking a queue manager offline from MSCS" on page 1233.
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 1226.

Delete a queue manager from a node

`hadltmqm.exe`

Consider the case where you have had a queue manager in your cluster, it has been moved from one node to another, and now you want to destroy it. Use the IBM MQ Explorer to delete it on the node where it currently is. The registry entries for it still exist on the other computer. To delete these, enter the following command at a prompt on that computer:

```
hadltmqm /m qmname
```

where `qmname` is the name of the queue manager to remove.

Check and save setup details

`amqmsysn.exe`

This utility presents a dialog showing full details of your IBM MQ MSCS Support setup, such as might be requested if you call IBM support. There is an option to save the details to a file.

Multi-instance queue managers

Multi

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 152 shows an example of a multi-instance configuration for queue manager QM1. IBM MQ is installed on two servers, one of which is a spare. One queue manager, QM1, has been created. One instance of QM1 is active, and is running on one server. The other instance of QM1 is running in standby on the other server, doing no active processing, but ready to take over from the active instance of QM1, if the active instance fails.

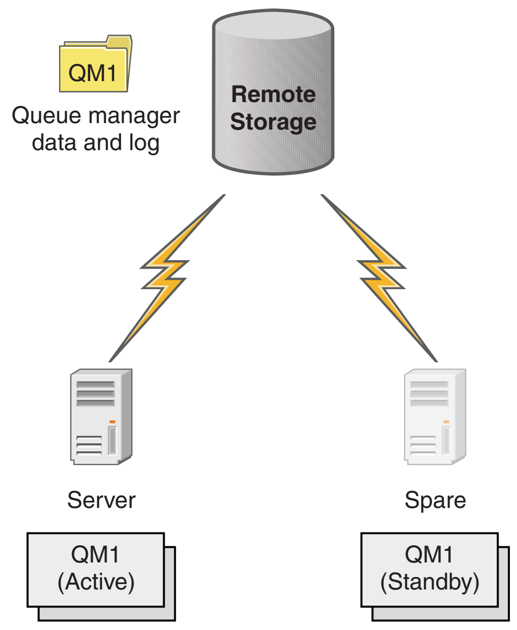


Figure 152. Multi-instance queue manager

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 non-z/OS platform supported by IBM WebSphere MQ Version 7.0.1 and later.

Automatic client reconnection is supported for MQI clients by IBM WebSphere MQ Version 7.0.1 and later.

Create a multi-instance queue manager:

Create a multi-instance queue manager, creating the queue manager on one server, and configuring IBM MQ on another server. Multi-instance queue managers share queue manager data and logs.

Most of the effort involved in creating a multi-instance queue manager is the task of setting up the shared queue manager data and log files. You must create shared directories on network storage, and make the directories available to other servers using network shares. These tasks need to be performed by someone with administrative authority, such as *root* on UNIX and Linux systems. The steps are as follows:

1. Create the shares for the data and log files.
2. Create the queue manager on one server.
3. Run the command **`dspmqinf`** on the first server to collect the queue manager configuration data and copy it into the clipboard.
4. Run the command **`addmqinf`** with the copied data to create the queue manager configuration on the second server.

You do not run **`crtmqm`** to create the queue manager again on the second server.

File access control

You need to take care that the user and group `mqm` on all other servers have permission to access the shares.

On UNIX and Linux, you need to make the `uid` and `gid` of `mqm` the same on all the systems. You might need to edit `/etc/passwd` on each system to set a common `uid` and `gid` for `mqm`, and then reboot your system.

On Microsoft Windows, the user ID that is running the queue manager processes must have full control permission to the directories containing the queue manager data and log files. You can configure the permission in two ways:

1. Create a queue manager with a global group as the alternative security principal. Authorize the global group to have full control access to the directories containing queue manager data and log files; see "Securing shared queue manager data and log directories and files on Windows" on page 1266. 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 1243, 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 1257, demonstrates how to set up a multi-instance queue manager using files secured in this way.

Configuration information

Configure as many queue manager instances as you need by modifying the IBM MQ queue manager configuration information about each server. Each server must have the same version of IBM MQ installed at a compatible fix level. The commands, **`dspmqinf`** and **`addmqinf`** assist you to configure the additional queue manager instances. Alternatively, you can edit the `mqs.ini` and `qm.ini` files directly. The topics, "Create a multi-instance queue manager on Linux" on page 1278, "Creating a multi-instance queue manager on domain workstations or servers on Windows" on page 1243, and "Creating a multi-instance queue manager on Windows domain controllers" on page 1257 are examples showing how to configure a multi-instance queue manager.

On Windows, UNIX and Linux systems, you can share a single `mqs.ini` file by placing it on the network share and setting the **`AMQ_MQS_INI_LOCATION`** environment variable to point to it.

Restrictions

1. Configure multiple instances of the same queue manager only on servers having the same operating system, architecture and endianness. For example, both machines must be either 32-bit or 64-bit.
2. All IBM MQ installations must be at release level 7.0.1 or higher.
3. Typically, active and standby installations are maintained at the same maintenance level. Consult the maintenance instructions for each upgrade to check whether you must upgrade all installations together.

   Note that the maintenance levels for the active and passive queue managers must be identical.
4. Share queue manager data and logs only between queue managers that are configured with the same IBM MQ user, group, and access control mechanism. IBM i For example, the network share set up on a Linux server could contain separate queue manager data and logs for UNIX and Linux queue managers, but could not contain the queue manager data used by IBM i.

   IBM i You can create multiple shares on the same networked storage for IBM i and for UNIX systems as long as the shares are different. You can give different shares different owners. The restriction is a consequence of the different names used for the IBM MQ users and groups between UNIX and IBM i. The fact that the user and group can have the same `uid` and `gid` does not relax the restriction.
5. On UNIX and Linux systems, configure the shared file system on networked storage with a `hard`, interruptible, mount rather than a `soft` mount. A hard interruptible mount forces the queue manager to hang until it is interrupted by a system call. Soft mounts do not guarantee data consistency after a server failure.
6. The shared log and data directories cannot be stored on a FAT, or an NFSv3 file system. For multi-instance queue managers on Windows, the networked storage must be accessed by the Common Internet File System (CIFS) protocol used by Windows networks.
7. z/OS z/OS does not support multi-instance queue managers. Use queue-sharing groups.

   Reconnectable clients do work with z/OS queue managers.

Windows domains and multi-instance queue managers: ▶ Windows

A multi-instance queue manager on Windows requires its data and logs to be shared. The share must be accessible to all instances of the queue manager running on different servers or workstations. Configure the queue managers and share as part of a Windows domain. The queue manager can run on a domain workstation or server, or on the domain controller.

Before configuring a multi-instance queue manager, read "Secure unshared queue manager data and log directories and files on Windows" on page 1269 and "Securing shared queue manager data and log directories and files on Windows" on page 1266 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 1243.

Run a multi-instance queue manager on domain workstations or servers

From Version 7.1, multi-instance queue managers run on a workstation or server that is a member of a domain. Before Version 7.1, multi-instance queue managers ran only on domain controllers; see "Run a multi-instance queue manager on domain controllers" on page 1243. To run a multi-instance queue manager on Windows, you require a domain controller, a file server, and two workstations or servers running the same queue manager connected to the same domain.

The change that makes it possible to run a multi-instance queue manager on any server or workstation in a domain, is that you can now create a queue manager with an additional security group. The additional security group is passed in the **crtmqm** command, in the -a parameter. You secure the directories that contain the queue manager data and logs with the group. The user ID that runs queue manager processes must be a member of this group. When the queue manager accesses the directories, Windows checks the permissions the user ID has to access the directories. By giving both the group and the user ID domain scope, the user ID running the queue manager processes has credentials from the global group. When the queue manager is running on a different server, the user ID running the queue manager processes can have the same credentials. The user ID does not have to be the same. It has to be a member of the alternative security group, as well as a member of the local mqm group.

The task of creating a multi-instance queue manager is the same as in Version 7.0.1 with one change. You must add the additional security group name to the parameters of the **crtmqm** command. The task is described in "Creating a multi-instance queue manager on domain workstations or servers on Windows" on page 1243.

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 1269. 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 1271 and "Reading and writing data and log files authorized by an alternative local security group" on page 1274. Another topic, "Securing shared queue manager data and log directories and files on Windows" on page 1266, 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 1246.
2. "Installing IBM MQ on a server or workstation in a Windows domain" on page 1249.
3. "Creating a shared directory for queue manager data and log files on Windows" on page 1252.
4. "Reading and writing shared data and log files authorized by an alternative global security group" on page 1254.

You can then do the task, “Creating a multi-instance queue manager on domain workstations or servers on Windows,” using the domain. Do these tasks to explore setting up a multi-instance queue manager before transferring your knowledge to a production domain.

Run a multi-instance queue manager on domain controllers

In Version 7.0.1, multi-instance queue managers ran only on domain controllers. Queue manager data could be secured with the domain `mqm` group. As the topic “Securing shared queue manager data and log directories and files on Windows” on page 1266 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 1257 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

Creating a multi-instance queue manager on domain workstations or servers on Windows: 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 UNIX and Linux. On Windows, the checks are always successful.

Do the steps in the following tasks. The tasks create the domain controller and domain, install IBM MQ for Windows on one server, and create the file share for data and log files. If you are configuring an existing domain controller, you might find it useful to try out the steps on a new Windows Server 2008. You can adapt the steps to your domain.

1. “Creating an Active Directory and DNS domain on Windows” on page 1246.
2. “Installing IBM MQ on a server or workstation in a Windows domain” on page 1249.

3. "Creating a shared directory for queue manager data and log files on Windows" on page 1252.
4. "Reading and writing shared data and log files authorized by an alternative global security group" on page 1254.

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 1249 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 1254, to create it
   b. Check the values of the `Prefix` and `InstallationName` parameters are correct for *venus*.

      On *mars*, run the **dspmqinf** command:

      ```
      dspmqinf QMGR
      ```

      The system response:

      ```
      QueueManager:
      Name=QMGR
      Directory=QMGR
      Prefix=C:\ProgramData\IBM\MQ
      DataPath=\\sun\wmq\data\QMGR
      InstallationName=Installation1
      ```
   c. Copy the machine-readable form of the **QueueManager** stanza to the clipboard.

      On *mars* run the **dspmqinf** command again, with the `-o command` parameter.

      ```
      dspmqinf -o command QMGR
      ```

      The system response:

      ```
      addmqinf -s QueueManager -v Name=QMGR
      -v Directory=QMGR -v Prefix="C:\ProgramData\IBM\MQ"
      -v DataPath=\\sun\wmq\data\QMGR
      ```
   d. On *venus* run the **addmqinf** command from the clipboard to create an instance of the queue manager on *venus*.

Adjust the command, if necessary, to accommodate differences in the `Prefix` or `InstallationName` parameters.

```
addmqinf -s QueueManager -v Name=QMGR
-v Directory=QMGR -v Prefix="C:\ProgramData\IBM\MQ"
-v DataPath=\\sun\wmq\data\QMGR
```

```
IBM MQ configuration information added.
```

3. Start the queue manager *QMGR* on *venus*, permitting standby instances.
   a. Check *QMGR* on *mars* is stopped.
      On *mars*, run the **dspmq** command:
      ```
      dspmq -m QMGR
      ```
      The system response depends on how the queue manager was stopped; for example:
      ```
      C:\Users\Administrator>dspmq -m QMGR
      QMNAME(QMGR) STATUS(Ended immediately)
      ```
   b. On *venus* run the **strmqm** command to start *QMGR* permitting standbys:
      ```
      strmqm -x QMGR
      ```
      The system response:
      ```
      IBM MQ queue manager 'QMGR' starting.
      The queue manager is associated with installation 'Installation1'.
      5 log records accessed on queue manager 'QMGR' during the log
      replay phase.
      Log replay for queue manager 'QMGR' complete.
      Transaction manager state recovered for queue manager 'QMGR'.
      IBM MQ queue manager 'QMGR' started using V7.1.0.0.
      ```

Results

To test the multi-instance queue manager switches over, do the following steps:

1. On *mars*, run the **strmqm** command to start *QMGR* permitting standbys:
   ```
   strmqm -x QMGR
   ```
   The system response:
   ```
   IBM MQ queue manager 'QMGR' starting.
   The queue manager is associated with installation 'Installation1'.
   A standby instance of queue manager 'QMGR' has been started.
   The active instance is running elsewhere.
   ```
2. On *venus* run the **endmqm** command:
   ```
   endmqm -r -s -i QMGR
   ```
   The system response on *venus*:
   ```
   IBM MQ queue manager 'QMGR' ending.
   IBM MQ queue manager 'QMGR' ending.
   IBM MQ queue manager 'QMGR' ending.
   IBM MQ queue manager 'QMGR' ending.
   IBM MQ queue manager 'QMGR' ending.
   IBM MQ queue manager 'QMGR' ending.
   IBM MQ queue manager 'QMGR' ended, permitting switchover to
   a standby instance.
   ```

And on *mars*:

```
dspmq
QMNAME(QMGR) STATUS(Running as standby)
C:\Users\wmquser2>dspmq
QMNAME(QMGR) STATUS(Running as standby)
C:\Users\wmquser2>dspmq
QMNAME(QMGR)  STATUS(Running)
```

What to do next

To verify a multi-instance queue manager using sample programs; see “Verifying the multi-instance queue manager on Windows” on page 1264.

Creating an Active Directory and DNS domain on Windows: 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 1242.

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*`\Adminstrator`.

    The server manager starts automatically.
11. Open the *wmq.example.com*`\Users` folder
    a. Open **Server Manager** > **Roles** > **Active Directory Domain Services** > ***wmq.example.com*** > **Users**.
12. Right-click **Users** > **New** > **Group**.
    a. Type a group name into the **Group name** field.

       Note: The preferred group name is `Domain mqm`. Type it exactly as shown.
       - Calling the group `Domain mqm` modifies the behavior of the "Prepare IBM MQ" wizard on a domain workstation or server. It causes the "Prepare IBM MQ" wizard automatically to add the group `Domain mqm` to the local `mqm` group on each new installation of IBM MQ in the domain.
       - You can install workstations or servers in a domain with no `Domain mqm` global group. If you do so, you must define a group with the same properties as `Domain mqm` group. You must make that group, or the users that are members of it, members of the local `mqm` group wherever IBM

MQ is installed in a domain. You can place domain users into multiple groups. Create multiple domain groups, each group corresponding to a set of installations that you want to manage separately. Split domain users, according to the installations they manage, into different domain groups. Add each domain group or groups to the local `mqm` group of different IBM MQ installations. Only domain users in the domain groups that are members of a specific local `mqm` group can create, administer, and run queue managers for that installation.

- The domain user that you nominate when installing IBM MQ on a workstation or server in a domain must be a member of the `Domain mqm` group, or of an alternative group you defined with same properties as the `Domain mqm` group.

b. Leave **Global** clicked as the **Group scope**, or change it to **Universal**. Leave **Security** clicked as the **Group type**. Click **OK**.

13. Add the rights, **Allow Read group membership** and **Allow Read groupMembershipSAM** to the rights of the `Domain mqm` global group.

a. In the Server Manager action bar, click **View** > **Advanced features**

b. In the Server Manager navigation tree, click **Users**

c. In the Users window, right-click **Domain mqm** > **Properties**

d. Click **Security** > **Advanced** > **Add...**. Type `Domain mqm` and click **Check names** > **OK**.

The **Name** field is prefilled with the string, `Domain mqm (`*`domain name`*`\Domain mqm)`.

e. Click **Properties**. In the **Apply to** list, select **Descendant User Objects**.

f. From the **Permissions** list, select the **Read group membership** and **Read groupMembershipSAM Allow** check boxes; click **OK** > **Apply** > **OK** > **OK**.

14. Add two or more users to the `Domain mqm` global group.

One user, *`wmquser1`* in the example, runs the IBM MQ service, and the other user, *`wmquser2`*, is used interactively.

A domain user is required to create a queue manager that uses the alternative security group in a domain configuration. It is not sufficient for the user ID to be an administrator, although an administrator has authority to run the **`crtmqm`** command. The domain user, who could be an administrator, must be a member of the local `mqm` group as well as of the alternative security group.

In the example, you make *`wmquser1`* and *`wmquser2`* members of the `Domain mqm` global group. The "Prepare IBM MQ" wizard automatically configures `Domain mqm` as a member of the local `mqm` group where ever the wizard is run.

You must provide a different user to run the IBM MQ service for each installation of IBM MQ on a single computer. You can reuse the same users on different computers.

a. In the Server Manager navigation tree, click **Users** > **New** > **User**

b. In the New Object - User window, type *`wmquser1`* into the **User logon name** field. Type *`WebSphere`* into the **First name** field, and *`MQ1`* into the **Last name** field. Click **Next**.

c. Type a password into the **Password** and **Confirm password** fields, and clear the **User must change password at next logon** check box. Click **Next** > **Finish**.

d. In the Users window, right-click ***WebSphere MQ*** > **Add to a group...**. Type `Domain mqm` and click **Check Names** > **OK** > **OK**.

e. Repeat steps a to d to add *`WebSphere MQ2`* as *`wmquser2`*.

15. Running IBM MQ as a service. If you need to run IBM MQ as a service, and then give the domain user (that you obtained from your domain administrator) the access to run as a service, carry out the following procedure:

a. Click **Start > Run...**. Type the command `secpol.msc` and click **OK**.

b. Open **Security Settings > Local Policies > User Rights Assignments**. In the list of policies, right-click **Log on as a service > Properties**.

c. Click **Add User or Group...** Type the name of the user you obtained from your domain administrator, and click **Check Names**

d. If prompted by a Windows Security window, type the user name and password of an account user or administrator with sufficient authority, and click **OK > Apply > OK**. Close the Local Security Policy window.

Note: On Windows Server 2008 and Windows Server 2012 the User Account Control (UAC) is enabled by default.

The UAC feature restricts the actions users can perform on certain operating system facilities, even if they are members of the Administrators group. You must take appropriate steps to overcome this restriction.

What to do next

Proceed to the next task, "Installing IBM MQ on a server or workstation in a Windows domain."

Installing IBM MQ on a server or workstation in a Windows domain: Windows

In this task, you install and configure IBM MQ on a server or workstation in the *wmq.example.com* Windows domain.

In a production scale configuration, you might have to tailor the configuration to an existing domain. For example, you might define different domain groups to authorize different shares, and to group the user IDs that run queue managers.

The example configuration consists of three servers:

sun A Windows Server 2008 domain controller. It owns the *wmq.example.com* domain that contains *Sun*, *mars*, and *venus*. For the purposes of illustration, it is also used as the file server.

mars A Windows Server 2008 used as the first IBM MQ server. It contains one instance of the multi-instance queue manager called *QMGR*.

venus A Windows Server 2008 used as the second IBM MQ server. It contains the second instance of the multi-instance queue manager called *QMGR*.

Replace the italicized names in the example, with names of your choosing.

Before you begin

1. Do the steps in "Creating an Active Directory and DNS domain on Windows" on page 1246 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 1242.

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 Version 6** from the list of items in the Local Area Connection Properties window. Click **Properties** > **Advanced...** and click the **DNS** tab.
   e. Under the DNS server addresses, click **Add...**.
   f. Type the IP address of the domain controller, which is also the DNS server, and click **Add**.
   g. Click **Append these DNS suffixes** > **Add...**.
   h. Type *wmq.example.com* and click **Add**.
   i. Type *wmq.example.com* in the **DNS suffix for this connection** field.
   j. Select **Register this connection's address in DNS** and **Use this connection's suffix in DNS registration**. Click **OK** > **OK** > **Close**
   k. Open a command window, and type the command **ipconfig /all** to review the TCP/IP settings.
2. On *mars*, add the computer to the *wmq.example.com* domain.
   a. Click **Start**
   b. Right-click **Computer** > **Properties**. In the Computer name, domain and workgroup settings division, click **Change settings**.
   c. In the System Properties windows, click **Change...**.
   d. Click Domain, type *wmq.example.com*, and click **OK**.
   e. Type the **User name** and **Password** of the domain controller administrator, who has the authority to permit the computer to join the domain, and click **OK**.
   f. Click **OK** > **OK** > **Close** > **Restart Now** in response to the "Welcome to the *wmq.example.com* domain" message.
3. Check that the computer is a member of the *wmq.example.com* domain
   a. On *sun*, log on to the domain controller as *wmq*\Administrator.
   b. Open **Server Manager** > **Active Directory Domain Services** > ***wmq.example.com*** > **Computers** and check *mars* is listed correctly in the Computers window.
4. Install IBM MQ for Windows on *mars*.

   For further information about running the IBM MQ for Windows installation wizard; see Installing IBM MQ server on Windows.
   a. On *mars*, log on as the local administrator, *mars*\Administrator.
   b. Run the **Setup** command on the IBM MQ for Windows installation media.

      The IBM MQ Launchpad application starts.
   c. Click **Software Requirements** to check that the prerequisite software is installed.
   d. Click **Network Configuration** > **Yes** to configure a domain user ID.

      The task, "Creating an Active Directory and DNS domain on Windows" on page 1246, configures a domain user ID for this set of tasks.
   e. Click **IBM MQ Installation**, select an installation language and click Launch IBM MQ Installer.
   f. Confirm the license agreement and click **Next** > **Next** > **Install** to accept the default configuration. Wait for the installation to complete, and click **Finish**.

You might want to change the name of the installation, install different components, configure a different directory for queue manager data and logs, or install into a different directory. If so, click **Custom** rather than **Typical**. IBM MQ is installed, and the installer starts the "Prepare IBM MQ" wizard.

Important: Do not run the wizard yet.

5. Configure the user that is going to run the IBM MQ service with the **Run as a service** right.

   Choose whether to configure the local `mqm` group, the `Domain mqm` group, or the user that is going to run the IBM MQ service with the right. In the example, you give the user the right.

   a. Click **Start** > **Run...**, type the command **`secpol.msc`** and click **OK**.

   b. Open **Security Settings** > **Local Policies** > **User Rights Assignments**. In the list of policies, right-click **Log on as a service** > **Properties**.

   c. Click **Add User or Group...** and type *`wmquser1`* and click **Check Names**

   d. Type the user name and password of a domain administrator, *`wmq`*`\Administrator`, and click **OK** > **Apply** > **OK**. Close the Local Security Policy window.

6. Run the "Prepare IBM MQ" wizard.

   For further information about running the "Prepare IBM MQ" wizard; see Configuring IBM MQ with the Prepare IBM MQ wizard.

   a. The IBM MQ Installer runs the "Prepare IBM MQ" automatically.

      To start the wizard manually, find the shortcut to the "Prepare IBM MQ" in the **Start** > **All programs** > **IBM MQ** folder. Select the shortcut that corresponds to the installation of IBM MQ in a multi-installation configuration.

   b. Click **Next** and leave **Yes** clicked in response to the question "Identify if there is a Windows 2000 or later domain controller in the network".

   c. Click **Yes** > **Next** in the first Configuring IBM MQ for Windows for Windows domain users window.

   d. In the second Configuring IBM MQ for Windows for Windows domain users window, type *`wmq`* in the **Domain** field. Type *`wmquser1`* in the **User name** field, and the password, if you set one, in the **Password** field. Click **Next**.

      The wizard configures and starts the IBM MQ with *wmquser1*.

   e. In the final page of the wizard, select or clear the check boxes as you require and click **Finish**.

What to do next

1. Do the task, "Reading and writing data and log files authorized by the local `mqm` group" on page 1271, 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 1252, to configure a file share to store the data and log files of a multi-instance queue manager.

Related information:
User rights required for an IBM MQ Windows Service

Creating a shared directory for queue manager data and log files on Windows: 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 1246, 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 1242.

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."

Reading and writing shared data and log files authorized by an alternative global security group: Windows

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 1246.
2. "Installing IBM MQ on a server or workstation in a Windows domain" on page 1249.
3. "Creating a shared directory for queue manager data and log files on Windows" on page 1252.

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 1242.

In this task, you create a queue manager that stores its data and logs in a remote directory on a file server. For the purposes of this example, the file server is the same server as the domain controller. The directory containing the data and log folders is shared with full control permission given to the global group `wmqha`.

Procedure

1. Log on to the domain server, *`mars`*, as the local administrator, *`mars`*`\Administrator`.
2. Open a command window.
3. Restart the IBM MQ service.

   You must restart the service so that the user ID it runs under acquires the additional security credentials you configured for it.

   Type the commands:

   ```
   endmqsvc
   strmqsvc
   ```

   The system responses:

   ```
   5724-H72 (C) Copyright IBM Corp. 1994, 2011.  ALL RIGHTS RESERVED.
   The MQ service for installation 'Installation1' ended successfully.
   ```

   And:

   ```
   5724-H72 (C) Copyright IBM Corp. 1994, 2011.  ALL RIGHTS RESERVED.
   The MQ service for installation 'Installation1' started successfully.
   ```
4. Create the queue manager.

   ```
   crtmqm -a wmq\wmqha -sax -u SYSTEM.DEAD.LETTER.QUEUE -md \\sun\wmq\data -ld \\sun\wmq\logs QMGR
   ```

   You must specify the domain, *`wmq`*, of the alternative security group *`wmqha`* by specifying full domain name of the global group *`"wmq\wmqha"`*.

   You must spell out the Universal Naming Convention (UNC) name of the share *`\\sun\wmq`*, and not use a mapped drive reference.

   The system response:

   ```
   IBM MQ queue manager created.
   Directory '\\sun\wmq\data\QMGR' created.
   The queue manager is associated with installation '1'
   Creating or replacing default objects for queue manager 'QMGR'
   Default objects statistics : 74 created. 0 replaced.
   Completing setup.
   Setup completed.
   ```

What to do next

Test the queue manager by putting and getting a message to a queue.

1. Start the queue manager.

   ```
   strmqm QMGR
   ```

   The system response:

```
IBM MQ queue manager 'QMGR' starting.
The queue manager is associated with installation '1'.
5 log records accessed on queue manager 'QMGR' during the log
replay phase.
Log replay for queue manager 'QMGR' complete.
Transaction manager state recovered for queue manager 'QMGR'.
IBM MQ queue manager 'QMGR' started using V7.1.0.0.
```

2. Create a test queue.

   ```
   echo define qlocal(QTEST) | runmqsc QMGR
   ```

   The system response:

   ```
   5724-H72 (C) Copyright IBM Corp. 1994, 2011.  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
```

Creating a multi-instance queue manager on Windows domain controllers: Windows

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 1243.

The example configuration consists of three servers:

sun A Windows Server 2008 server used as the first domain controller. It defines the *wmq.example.com* domain that contains *sun*, *earth*, and *mars*. It contains one instance of the multi-instance queue manager called *QMGR*.

earth A Windows Server 2008 used as the second domain controller IBM MQ server. It contains the second instance of the multi-instance queue manager called *QMGR*.

mars A Windows Server 2008 used as the file server.

Replace the italicized names in the example, with names of your choosing.

Before you begin

1. On Windows, you do not need to verify the file system that you plan to store queue manager data and log files on. The checking procedure, Verifying shared file system behavior, is applicable to UNIX and Linux. On Windows, the checks are always successful.
2. Do the steps in "Creating an Active Directory and DNS domain on Windows" on page 1246 to create the first domain controller.
3. Do the steps in "Adding a second Windows domain controller to an example domain" on page 1260 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 1262 to install IBM MQ on the two domain controllers.

About this task

On a file server in the same domain create a share for the queue manager log and data directories. Next, create the first instance of a multi-instance queue manager that uses the file share on one of the domain controllers. Create the other instance on the other domain controller and finally verify the configuration. You can create the file share on a domain controller.

In the sample, *sun* is the first domain controller, *earth* the second, and *mars* is the file server.

Procedure

1. Create the directories that are to contain the queue manager data and log files.
   a. On *mars*, type the command:
   ```
   md c:\wmq\data , c:\wmq\logs
   ```
2. Share the directories that are to contain the queue manager data and log files.

You must permit full control access to the domain local group `mqm`, and the user ID you use to create the queue manager. In the example, user IDs that are members of `Domain Administrators` have the authority to create queue managers.

The file share must be on a server that is in the same domain as the domain controllers. In the example, the server *`mars`* is in the same domain as the domain controllers.

a. In Windows Explorer, right-click *c:\wmq* > **Properties**.
b. Click the **Security** tab and click **Advanced** > **Edit...**.
c. Clear the check box for **Include inheritable permissions from this object's owner**. Click **Copy** in the Windows Security window.
d. Select the lines for Users in the list of **Permission entries** and click **Remove**. Leave the lines for SYSTEM, Administrators, and CREATOR OWNER in the list of **Permission entries**.
e. Click **Add...**, and type the name of the domain local group *`mqm`*. Click **Check Names**
f. In response to a Windows Security window, Type the name and password of the `Domain Administrator` and click **OK** > **OK**.
g. In the Permission Entry for `wmq` window, select **Full Control** in the list of **Permissions**.
h. Click **OK** > **Apply** > **OK** > **OK** > **OK**
i. Repeat steps e to h to add `Domain Administrators`.
j. In Windows Explorer, right-click *c:\wmq* > **Share...**.
k. Click **Advanced Sharing...** and select the **Share this folder** check box. Leave the share name as *`wmq`*.
l. Click **Permissions** > **Add...**, and type the name of the domain local group *`mqm`* **;** `Domain Administrators`. Click **Check Names**.
m. In response to a Windows Security window, Type the name and password of the `Domain Administrator` and click **OK** > **OK**.

3. Create the queue manager *`QMGR`* on the first domain controller, *`sun`*.

   ```
   crtmqm -sax -u SYSTEM.DEAD.LETTER.QUEUE -md \\mars\wmq\data -ld \\mars\wmq\logs QMGR
   ```

   The system response:

   ```
   IBM MQ queue manager created.
   Directory '\\mars\wmq\data\QMGR' created.
   The queue manager is associated with installation 'Installation1'.
   Creating or replacing default objects for queue manager 'QMGR'.
   Default objects statistics : 74 created. 0 replaced. 0 failed.
   Completing setup.
   Setup completed.
   ```

4. Start the queue manager on *`sun`*, permitting a standby instance.

   ```
   strmqm -x QMGR
   ```

   The system response:

   ```
   IBM MQ queue manager 'QMGR' starting.
   The queue manager is associated with installation 'Installation1'.
   5 log records accessed on queue manager 'QMGR' during the log
   replay phase.
   Log replay for queue manager 'QMGR' complete.
   Transaction manager state recovered for queue manager 'QMGR'.
   IBM MQ queue manager 'QMGR' started using V7.1.0.0.
   ```

5. Create a second instance of *`QMGR`* on *`earth`*.

   a. Check the values of the `Prefix` and `InstallationName` parameters are correct for *`earth`*.

      On *`sun`*, run the **`dspmqinf`** command:

      ```
      dspmqinf QMGR
      ```

The system response:

```
QueueManager:
Name=QMGR
Directory=QMGR
Prefix=C:\ProgramData\IBM\MQ
DataPath=\\mars\wmq\data\QMGR
InstallationName=Installation1
```

b. Copy the machine-readable form of the **QueueManager** stanza to the clipboard.

   On *sun* run the **dspmqinf** command again, with the `-o command` parameter.

   ```
   dspmqinf -o command QMGR
   ```

   The system response:

   ```
   addmqinf -s QueueManager -v Name=QMGR
   -v Directory=QMGR -v Prefix="C:\ProgramData\IBM\MQ"
   -v DataPath=\\mars\wmq\data\QMGR
   ```

c. On *earth* run the **addmqinf** command from the clipboard to create an instance of the queue manager on *earth*.

   Adjust the command, if necessary, to accommodate differences in the `Prefix` or `InstallationName` parameters.

   ```
   addmqinf -s QueueManager -v Name= QMGR
   -v Directory= QMGR -v Prefix="C:\Program Files\IBM\WebSphere MQ"
   -v DataPath=\\mars\wmq\data\QMGR

   IBM MQ configuration information added.
   ```

6. Start the standby instance of the queue manager on *earth*.

   ```
   strmqm -x QMGR
   ```

   The system response:

   ```
   IBM MQ queue manager 'QMGR' starting.
   The queue manager is associated with installation 'Installation1'.
   A standby instance of queue manager 'QMGR' has been started. The active
   instance is running elsewhere.
   ```

Results

Verify that the queue manager switches over from *sun* to *earth*:

1. On *sun*, run the command:

   ```
   endmqm -i -r -s QMGR
   ```

   The system response on *sun*:

   ```
   IBM MQ queue manager 'QMGR' ending.
   IBM MQ queue manager 'QMGR' ending.
   IBM MQ queue manager 'QMGR' ending.
   IBM MQ queue manager 'QMGR' ending.
   IBM MQ queue manager 'QMGR' ending.
   IBM MQ queue manager 'QMGR' ending.
   IBM MQ queue manager 'QMGR' ended, permitting switchover to
   a standby instance.
   ```

2. On *earth* repeatedly type the command:

   ```
   dspmq
   ```

The system responses:

```
QMNAME(QMGR) STATUS(Running as standby)
QMNAME(QMGR) STATUS(Running as standby)
QMNAME(QMGR) STATUS(Running)
```

What to do next

To verify a multi-instance queue manager using sample programs; see "Verifying the multi-instance queue manager on Windows" on page 1264.

Related tasks:

"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.

"Installing IBM MQ on Windows domain controllers in an example domain" on page 1262
Put your short description here; used for first paragraph and abstract.

Related information:

Windows 2000, Windows Server 2003, and Windows Server 2008 cluster nodes as domain controllers

Adding a second Windows domain controller to an example domain: Windows

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 1246 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 1242.

Procedure

1. Add the domain controller, *sun.wmq.example.com* to *earth* as a DNS server.
   a. On *earth*, log on as *earth*\Administrator and click **Start**.
   b. Right-click **Network** > **Properties** > **Manage network connections**.
   c. Right-click the network adapter, click **Properties**.

      The system responds with the Local Area Connection Properties window listing items the connection uses.
   d. Select the **Internet Protocol Version 4** or **Internet Protocol Version 6** from the list of items in the Local Area Connection Properties window. Click **Properties** > **Advanced...** and click the **DNS** tab.
   e. Under the DNS server addresses, click **Add...**.
   f. Type the IP address of the domain controller, which is also the DNS server, and click **Add**.
   g. Click **Append these DNS suffixes** > **Add...**.
   h. Type *wmq.example.com* and click **Add**.
   i. Type *wmq.example.com* in the **DNS suffix for this connection** field.
   j. Select **Register this connection's address in DNS** and **Use this connection's suffix in DNS registration**. Click **OK** > **OK** > **Close**
   k. Open a command window, and type the command **ipconfig /all** to review the TCP/IP settings.
2. Log on to the domain controller, *sun*, as the local or Workgroup administrator.

   If the server is already configured as a domain controller, you must log on as a domain administrator.
3. Run the Active Directory Domain Services wizard.
   a. Click **Start** > **Run...** Type dcpromo and click **OK**.

   If the Active Directory binary files are not already installed, Windows installs the files automatically.
4. Configure *earth* as the second domain controller in the *wmq.example.com* domain.
   a. In the first window of the wizard, leave the **Use advanced mode installation** check box clear. Click **Next** > **Next** and click **Create Add a domain controller to an existing domain** > **Next**.
   b. Type *wmq* into the **Type the name of any domain in this forest ...** field. The **Alternate credentials** radio button is clicked, click **Set...**. Type in the name and password of the domain administrator and click **OK** > **Next** > **Next** > **Next**.
   c. In the Additional Domain Controller Options window accept the **DNS server** and **Global catalog** options, which are selected; click **Next** > **Next**.
   d. On the Directory Services Restore Mode Administrator Password, type in a **Password** and **Confirm password** and click **Next** > **Next**.
   e. When prompted for **Network Credentials**, type in the password of the domain administrator. Select **Reboot on completion** in the final wizard window.
   f. After a while, a window might open with a **DCPromo** error concerning DNS delegation; click **OK**. The server reboots.

Results

When *earth* has rebooted, log on as Domain Administrator. Check that the wmq.example.com domain has been replicated to *earth*.

What to do next

Continue with installing IBM MQ ; see "Installing IBM MQ on Windows domain controllers in an example domain" on page 1262.

Related tasks:

"Creating an Active Directory and DNS domain on Windows" on page 1246
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.

Installing IBM MQ on Windows domain controllers in an example domain: Windows

Install and configure installations of IBM MQ on both domain controllers in the *wmq.example.com* domain.

Put your short description here; used for first paragraph and abstract.

The example configuration consists of three servers:

sun A Windows Server 2008 server used as the first domain controller. It defines the *wmq.example.com* domain that contains *sun*, *earth*, and *mars*. It contains one instance of the multi-instance queue manager called *QMGR*.

earth A Windows Server 2008 used as the second domain controller IBM MQ server. It contains the second instance of the multi-instance queue manager called *QMGR*.

mars A Windows Server 2008 used as the file server.

Replace the italicized names in the example, with names of your choosing.

Before you begin

1. Do the steps in "Creating an Active Directory and DNS domain on Windows" on page 1246 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 1260 to create a second domain controller, *earth*, for the domain *wmq.example.com*. Change the italicized names to suit your configuration.
3. See Hardware and software requirements on Windows systems for other Windows versions you can run IBM MQ on.

About this task

Install and configure installations of IBM MQ on both domain controllers in the *wmq.example.com* domain.

Procedure

1. Install IBM MQ on *sun* and *earth*.

   For further information about running the IBM MQ for Windows installation wizard; see Installing IBM MQ server on Windows.

   a. On both *sun* and *earth*, log on as the domain administrator, *wmq*\Administrator.
   b. Run the **Setup** command on the IBM MQ for Windows installation media.

      The IBM MQ Launchpad application starts.
   c. Click **Software Requirements** to check that the prerequisite software is installed.
   d. Click **Network Configuration** > **No**.

      You can configure either a domain user ID or not for this installation. The user ID that is created is a domain local user ID.
   e. Click **IBM MQ Installation**, select an installation language and click Launch IBM MQ Installer.
   f. Confirm the license agreement and click **Next** > **Next** > **Install** to accept the default configuration. Wait for the installation to complete, and click **Finish**.

If you want to change the name of the installation, install different components, configure a different directory for queue manager data and logs, or install into a different directory, click **Custom** rather than **Typical**. IBM MQ is installed, and the installer starts the “Prepare IBM MQ” wizard.

The IBM MQ for Windows installation configures a domain local group `mqm`, and a domain group `Domain mqm`. It makes `Domain mqm` a member of `mqm`. Subsequent domain controllers in the same domain share the `mqm` and `Domain mqm` groups.

2. On both *`earth`* and *`sun`*, run the “Prepare IBM MQ” wizard.

   For further information about running the “Prepare IBM MQ” wizard, see Configuring IBM MQ with the Prepare IBM MQ wizard.

   a. The IBM MQ installer runs the “Prepare IBM MQ” automatically.

      To start the wizard manually, find the shortcut to the “Prepare IBM MQ” in the **Start** > **All programs** > **IBM MQ** folder. Select the shortcut that corresponds to the installation of IBM MQ in a multi-installation configuration.

   b. Click **Next** and leave **No** clicked in response to the question “Identify if there is a Windows 2000 or later domain controller in the network”[12] .

   c. In the final page of the wizard, select or clear the check boxes as you require and click **Finish**.

   The “Prepare IBM MQ” wizard creates a domain local user `MUSR_MQADMIN` on the first domain controller, and another domain local user `MUSR_MQADMIN1` on the second domain controller. The wizard creates the IBM MQ service on each controller, with `MUSR_MQADMIN` or `MUSR_MQADMIN1` as the user that logs on the service.

3. Define a user that has permission to create a queue manager.

   The user must have the right to log on locally, and be a member of the domain local `mqm` group. On domain controllers, domain users do not have the right to log on locally, but administrators do. By default, no user has both these attributes. In this task, add domain administrators to the domain local `mqm` group.

   a. Open **Server Manager** > **Roles** > **Active Directory Domain Services** > ***wmq.example.com*** > **Users**.

   b. Right-click **Domain Admins** > **Add to a group...** and type `mqm` ; click **Check names** > **OK** > **OK**

Results

1. Check that the “Prepare IBM MQ” created the domain user, `MUSR_MQADMIN`:

   a. Open **Server Manager** > **Roles** > **Active Directory Domain Services** > ***wmq.example.com*** > **Users**.

   b. Right-click **MUSR_MQADMIN** > **Properties...** > **Member Of**, and see that it is a member of `Domain users` and `mqm`.

2. Check that `MUSR_MQADMIN` has the right to run as a service:

   a. Click Click **Start** > **Run...**, type the command **`secpol.msc`** and click **OK**.

   b. Open **Security Settings** > **Local Policies** > **User Rights Assignments**. In the list of policies, right-click **Log on as a service** > **Properties**, and see `MUSR_MQADMIN` is listed as having the right to log on as a service. Click **OK**.

What to do next

1. Do the task, “Reading and writing data and log files authorized by the local `mqm` group” on page 1271, 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 1257, to complete the task of configuring a multi-instance queue manager on domain controllers.

12. 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.

Related information:

User rights required for an IBM MQ Windows Service

Verifying the multi-instance queue manager on Windows: 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 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 1257. 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) +
CONNAME(' ipaddr1 (1414), ipaddr2 (1414)') QMNAME(QM1) REPLACE
```

```
START CHANNEL(CHANNEL1)
DEFINE LISTENER(LISTENER.TCP) TRPTYPE(TCP) CONTROL(QMGR)
DISPLAY LISTENER(LISTENER.TCP) CONTROL
DISPLAY LSSTATUS(LISTENER.TCP) STATUS
```

Note: By leaving the **MCAUSER** parameter blank, the client user ID is sent to the server. The client user ID must have the correct permissions on the servers. An alternative is to set the **MCAUSER** parameter in the SVRCONN channel to the user ID you have configured on the server.

2. Open a command prompt with the path N:\ and run the command:

```
runmqsc -m QM1 < hasample.tst
```

3. Verify that the listener is running and has queue manager control, either by inspecting the output of the **runmqsc** command.

```
LISTENER(LISTENER.TCP)CONTROL(QMGR)
LISTENER(LISTENER.TCP)STATUS(RUNNING)
```

Or, using the IBM MQ Explorer that the TCPIP listener is running and has Control = Queue Manager.

On the client

1. Map the shared directory C:\MQHA on the server to N:\ on the client.
2. Open a command prompt with the path N:\. Set the environment variable MQCHLLIB to point to the client channel definition table (CCDT) on the server:

```
SET MQCHLLIB=N:\data\QM1\@ipcc
```

3. At the command prompt type the commands:

```
start amqsghac TARGET QM1
start amqsmhac -s SOURCE -t TARGET -m QM1
start amqsphac SOURCE QM1
```

Note: If you have problems, start the applications at a command prompt so that the reason code is printed out on the console, or look at the AMQERR01.LOG file in the N:\data\QM1\errors folder.

On the server running the active queue manager instance

1. Either:
   - Close the window running the VMware image with the active server instance.
   - Using the IBM MQ Explorer, stop the active queue manager instance, allowing switchover to the standby instance and instructing re-connectable clients to reconnect.
2. The three clients eventually detect the connection is broken, and then reconnect. In this configuration, if you close the server window, it is taking about seven minutes for all three connections to be reestablished. Some connections are reestablished well before others.

Results

```
N:\>amqsphac SOURCE QM1
Sample AMQSPHAC start
target queue is SOURCE
message Message 1
message Message 2
message Message 3
message Message 4
message Message 5
17:05:25 : EVENT : Connection Reconnecting (Delay: 0ms)
17:05:47 : EVENT : Connection Reconnecting (Delay: 0ms)
17:05:52 : EVENT : Connection Reconnected
message Message 6
message Message 7
message Message 8
message Message 9
```

```
N:\>amqsmhac -s SOURCE -t TARGET -m QM1
Sample AMQSMHA0 start

17:05:25 : EVENT : Connection Reconnecting (Delay: 97ms)
17:05:48 : EVENT : Connection Reconnecting (Delay: 0ms)
17:05:53 : EVENT : Connection Reconnected
```

```
N:\>amqsghac TARGET QM1
Sample AMQSGHAC start
message Message 1
message Message 2
message Message 3
message Message 4
message Message 5
17:05:25 : EVENT : Connection Reconnecting (Delay: 156ms)
17:05:47 : EVENT : Connection Reconnecting (Delay: 0ms)
17:05:52 : EVENT : Connection Reconnected
message Message 6
message Message 7
message Message 8
message Message 9
```

Securing shared queue manager data and log directories and files on Windows: 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 153 on page 1267. 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.

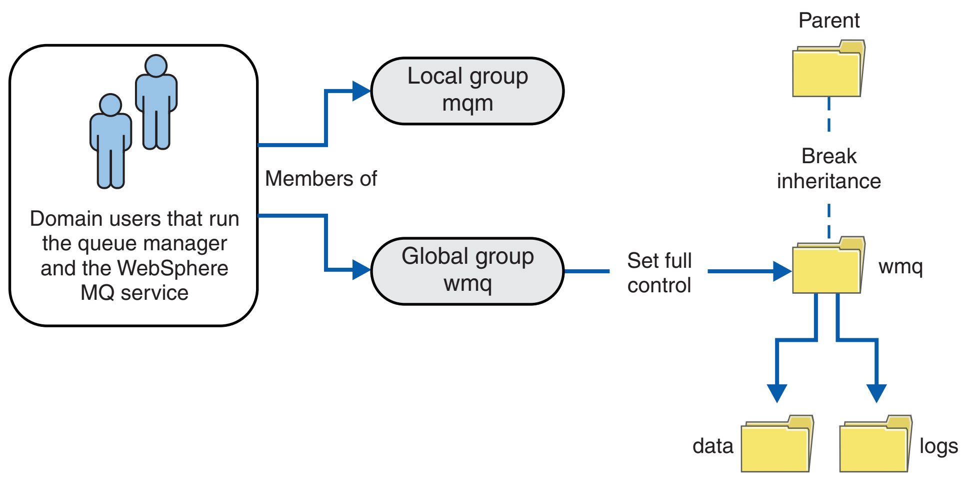


Figure 153. Securing queue manager data and logs using an alternative global security group (1)

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 153, 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 153 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 154 on page 1268. 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.

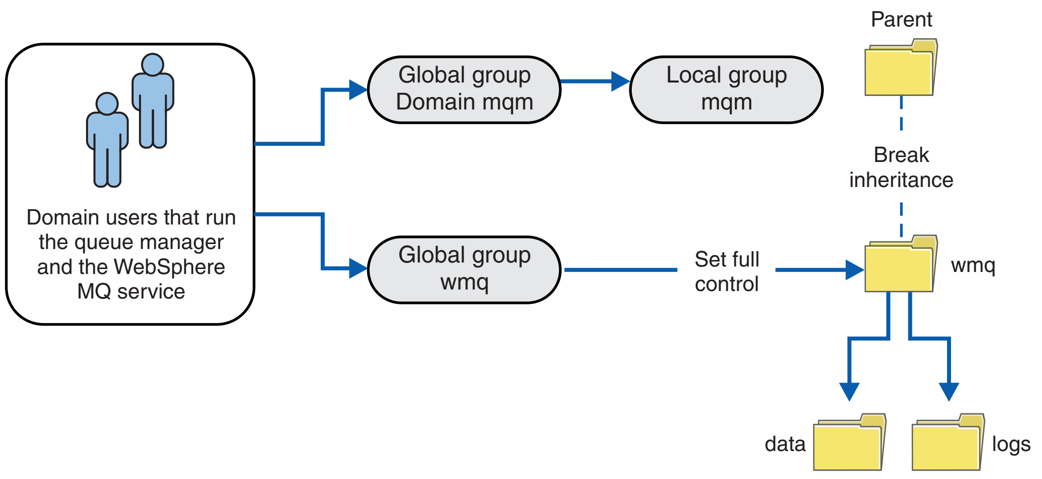


Figure 154. Securing queue manager data and logs using an alternative global security group (2)

The organization in Figure 154 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 155.

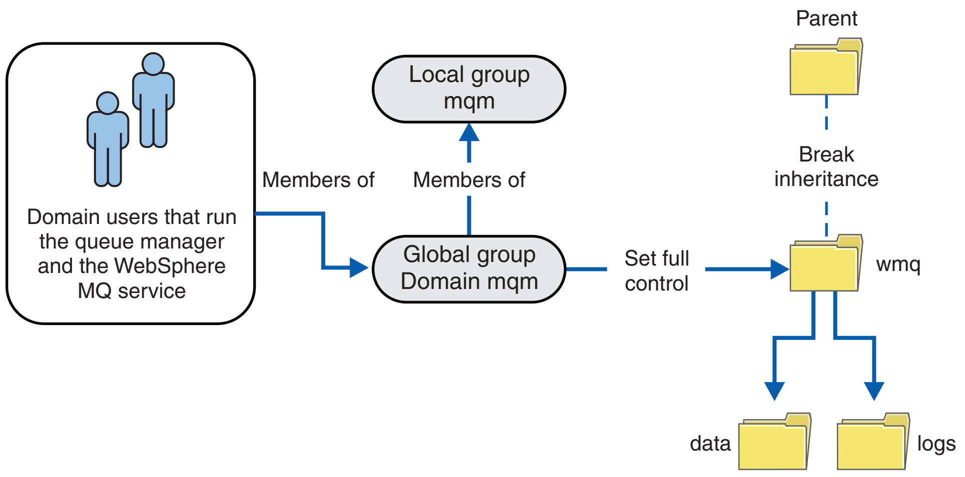


Figure 155. Securing queue manager data and logs using an alternative global security group (3)

Alternatively, you might need a finer degree of access control, with different queue managers restricted to being able to access different folders; see Figure 156 on page 1269. In Figure 156 on page 1269, 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 155, because there are two groups of users.

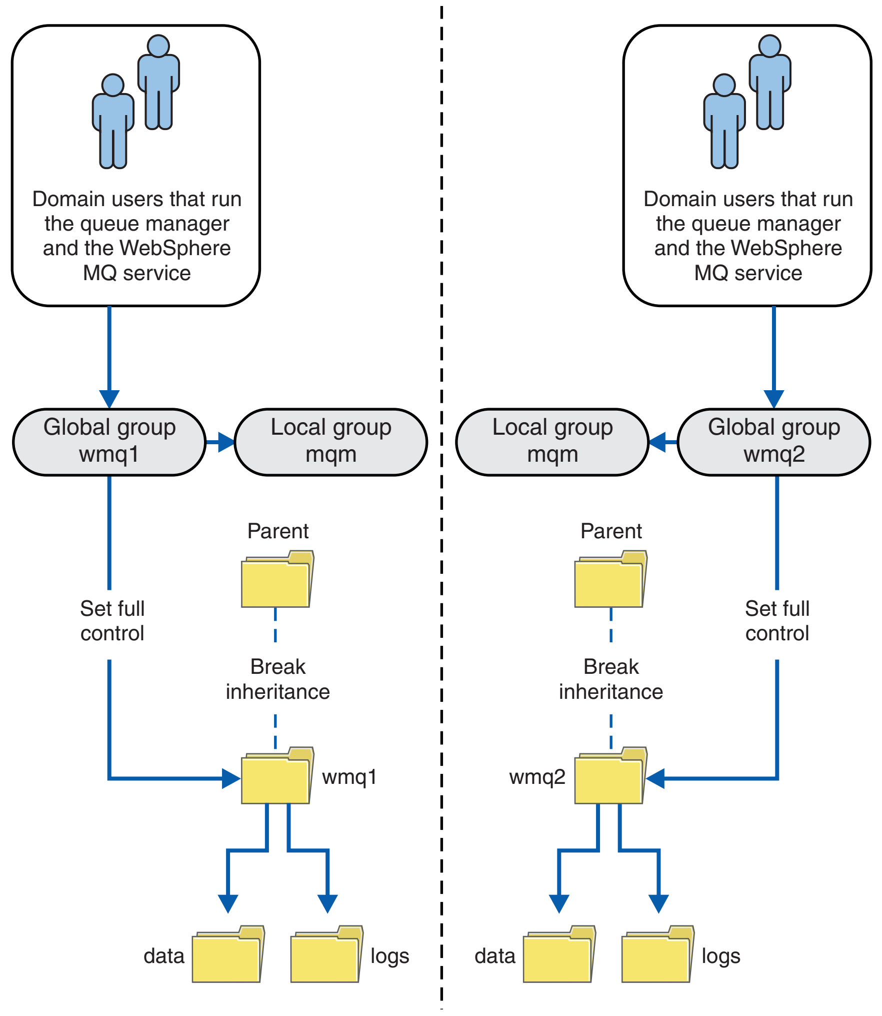


Figure 156. Securing queue manager data and logs using an alternative global security principal (4)

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 155 on page 1268.

Secure unshared queue manager data and log directories and files on Windows: 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 1271, 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 1274, 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 1266.

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 1271, 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 157. The local group, `wmq`, is an example of an alternative security group.

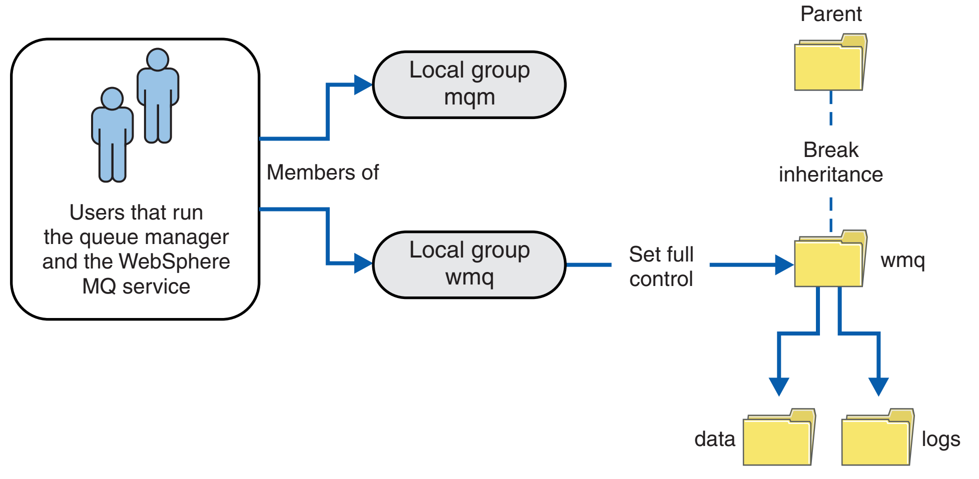


Figure 157. Securing queue manager data and logs using an alternative local security group, `wmq`

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 1266.
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: Windows

The task illustrates how to create a queue manager with its data and logs files stored in any directory of your choosing. Access to the files is secured by the local `mqm` group. The directory is not shared.

Before you begin

1. Install IBM MQ for Windows as the primary installation.
2. Run the "Prepare IBM MQ" wizard. For this task, configure the installation either to run with a local user ID, or a domain user ID. Eventually, to complete all the tasks in "Windows domains and multi-instance queue managers" on page 1242, 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 1242.

On Windows, you can create the default data and log paths for an IBM MQ for Windows in any directories of your choosing. The installation and configuration wizard automatically gives the local `mqm` group, and the user ID that is running the queue manager processes, access to the directories. If you create a queue manager specifying different directories for queue manager data and log files, you must configure full control permission to the directories.

In this example, you give the queue manager full control over its data and log files by giving the local `mqm` group permission to the directory *`c:\wmq`*.

The **`crtmqm`** command creates a queue manager that starts automatically when the workstation starts using the IBM MQ service.

The task is illustrative; it uses specific values that you can change. The values you can change are in italics. At the end of the task, follow the instructions to remove all the changes you made.

Procedure

1. Open a command prompt.
2. Type the command:

```
md c:\wmq\data, c:\wmq\logs
```

3. Set the permissions on the directories to allow the local mqm group read and write access.

   ```
   cacls c:\wmq/T /E /G mqm:F
   ```

   The system response:

   ```
   processed dir: c:\wmq
   processed dir: c:\wmq\data
   processed dir: c:\wmq\logs
   ```

4. Optional: Switch to a user ID that is a member of the local mqm group.

   You can continue as Administrator, but for a realistic production configuration, continue with a user ID with more restricted rights. The user ID must at least be a member of the local mqm group.

   If the IBM MQ installation is configured as part of a domain, make the user ID a member of the Domain mqm group. The "Prepare IBM MQ" wizard makes the Domain mqm global group a member of the local mqm group, so you do not have to make the user ID directly a member of the local mqm group.

5. Create the queue manager.

   ```
   crtmqm -sax -u SYSTEM.DEAD.LETTER.QUEUE -md c:\wmq\data -ld c:\wmq\logs QMGR
   ```

   The system response:

   ```
   IBM MQ queue manager created.
   Directory 'c:\wmq\data\QMGR' created.
   The queue manager is associated with installation '1'
   Creating or replacing default objects for queue manager 'QMGR'
   Default objects statistics : 74 created. 0 replaced.
   Completing setup.
   Setup completed.
   ```

6. Check that the directories created by the queue manager are in the *c:\wmq* directory.

   ```
   dir c:\wmq/D /B /S
   ```

7. Check that the files have read and write, or full control permission for the local mqm group.

   ```
   cacls c:\wmq\*.*
   ```

What to do next

Test the queue manager by putting and getting a message to a queue.

1. Start the queue manager.

   ```
   strmqm QMGR
   ```

   The system response:

   ```
   IBM MQ queue manager 'QMGR' starting.
   The queue manager is associated with installation '1'.
   5 log records accessed on queue manager 'QMGR' during the log
   replay phase.
   Log replay for queue manager 'QMGR' complete.
   Transaction manager state recovered for queue manager 'QMGR'.
   IBM MQ queue manager 'QMGR' started using V7.1.0.0.
   ```

2. Create a test queue.

   ```
   echo define qlocal(QTEST) | runmqsc QMGR
   ```

   The system response:

```
5724-H72 (C) Copyright IBM Corp. 1994, 2011.  ALL RIGHTS RESERVED.
Starting MQSC for queue manager QMGR.


1 : define qlocal(QTEST)
AMQ8006: IBM MQ queue created.
One MQSC command read.
No commands have a syntax error.
All valid MQSC commands were processed.
```

3. Put a test message using the sample program **amqsput**.

   ```
   echo 'A test message' | amqsput QTEST QMGR
   ```

   The system response:

   ```
   Sample AMQSPUT0 start
   target queue is QTEST
   Sample AMQSPUT0 end
   ```

4. Get the test message using the sample program **amqsget**.

   ```
   amqsget QTEST QMGR
   ```

   The system response:

   ```
   Sample AMQSGET0 start
   message A test message
   Wait 15 seconds ...
   no more messages
   Sample AMQSGET0 end
   ```

5. Stop the queue manager.

   ```
   endmqm -i QMGR
   ```

   The system response:

   ```
   IBM MQ queue manager 'QMGR' ending.
   IBM MQ queue manager 'QMGR' ended.
   ```

6. Delete the queue manager.

   ```
   dltmqm QMGR
   ```

   The system response:

   ```
   IBM MQ queue manager 'QMGR' deleted.
   ```

7. Delete the directories you created.

   Tip: Add the /Q option to the commands to prevent the command prompting to delete each file or directory.

   ```
   del /F /S C:\wmq\*.*
   rmdir /S C:\wmq
   ```

Related concepts:

"Windows domains and multi-instance queue managers" on page 1242
A multi-instance queue manager on Windows requires its data and logs to be shared. The share must be accessible to all instances of the queue manager running on different servers or workstations. Configure the queue managers and share as part of a Windows domain. The queue manager can run on a domain workstation or server, or on the domain controller.

Related tasks:

"Reading and writing shared data and log files authorized by an alternative global security group" on page 1254
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.

"Creating a multi-instance queue manager on domain workstations or servers on Windows" on page 1243
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.

Reading and writing data and log files authorized by an alternative local security group: Windows

This task shows how to use the `-a` flag on the **crtmqm** command. The flag provides the queue manager with an alternative local security group to give it access to its log and data files.

Before you begin

1. Install IBM MQ for Windows as the primary installation.
2. Run the "Prepare IBM MQ" wizard. For this task, configure the installation either to run with a local user ID, or a domain user ID. Eventually, to complete all the tasks in "Windows domains and multi-instance queue managers" on page 1242, 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 1242.

On Windows, you can create the default data and log paths for an IBM MQ for Windows in any directories of your choosing. The installation and configuration wizard automatically gives the local `mqm` group, and the user ID that is running the queue manager processes, access to the directories. If you create a queue manager specifying different directories for queue manager data and log files, you must configure full control permission to the directories.

In this example, you provide the queue manager with an alternative security local group that has full control authorization to the directories. The alternative security group gives the queue manager permission to manage files in the directory. The primary purpose of the alternate security group is to authorize an alternate security global group. Use an alternate security global group to set up a multi-instance queue manager. In this example, you configure a local group to familiarize yourself with the use of an alternate security group without installing IBM MQ in a domain. It is unusual to configure a local group as an alternative security group.

The **crtmqm** command creates a queue manager that starts automatically when the workstation starts using the IBM MQ service.

The task is illustrative; it uses specific values that you can change. The values you can change are in italics. At the end of the task, follow the instructions to remove all the changes you made.

Procedure

1. Set up an alternative security group.

   The alternative security group is typically a domain group. In the example, you create a queue manager that uses a local alternate security group. With a local alternate security group, you can do the task with an IBM MQ installation that is not part of a domain.

   a. Run the **`lusrmgr.msc`** command to open the Local Users and Groups window.

   b. Right-click **Groups** > **New Group...**

   c. In the **Group name** field, type *`altmqm`* and click **Create** > **Close**.

   d. Identify the user ID that runs the IBM MQ service.

      1) Click **Start** > **Run...**, type `services.msc` and click **OK**.

      2) Click the IBM MQ service in the list of services, and click the Log On tab.

      3) 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 1243, the domain user IDs created in “Creating an Active Directory and DNS domain on Windows” on page 1246 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.

      1) In the list of groups right-click **altmqm** > **Properties** > **Add...**.

      2) In the Select Users, Computers, or Groups window type *`wmquser1`* **;** *`wmquser2`* and click **Check Names**.

      3) 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, 2011.  ALL RIGHTS RESERVED.
   The MQ service for installation 'Installation1' ended successfully.
   ```

   And:

```
5724-H72 (C) Copyright IBM Corp. 1994, 2011.  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, 2011.  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
   ```

Create a multi-instance queue manager on Linux: Linux

An example shows how to set up a multi-instance queue manager on Linux. The setup is small to illustrate the concepts involved. The example is based on Linux Red Hat Enterprise 5. The steps differ on other UNIX platforms.

About this task

The example is set up on a 2 GHz notebook computer with 3 GB RAM running Windows 7 Service Pack 1. Two VMware virtual machines, Server1 and Server2, run Linux Red Hat Enterprise 5 in 640 MB images. Server1 hosts the network file system (NFS), the queue manager logs and an HA instance. It is not usual practice for the NFS server also to host one of the queue manager instances; this is to simplify the example. Server2 mounts Server1's queue manager logs with a standby instance. A WebSphere MQ MQI client is installed on an additional 400 MB VMware image that runs Windows 7 Service Pack 1 and runs the sample high availability applications. All the virtual machines are configured as part of a VMware host-only network for security reasons.

Note: You should put only queue manager data on an NFS server. On the NFS, use the following three options with the mount command to make the system secure:

- **noexec**
  By using this option, you stop binary files from being run on the NFS, which prevents a remote user from running unwanted code on the system.
- **nosuid**
  By using this option, you prevent the use of the set-user-identifier and set-group-identifier bits, which prevents a remote user from gaining higher privileges.
- **nodev** By using this option, you stop character and block special devices from being used or defined, which prevents a remote user from getting out of a chroot jail.

Procedure

1. Log in as root.
2. Read Installing IBM MQ - 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:
      1) **`mkdir`** *`/MQHA`*
      2) **`mkdir`** *`/MQHA/logs`*
      3) **`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:
      1) **`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 [13] /MQHA *(rw,sync,no_wdelay,fsid=0) to /etc/exports
9. For Server1, complete the following steps:
   a. Start the NFS daemon: */etc/init.d/* **nfs** start
   b. Copy the queue manager configuration details from Server1:

      ```
      dspmqinf -o command QM1
      ```

      and copy the result to the clipboard:

      ```
      addmqinf -s QueueManager
      -v Name=QM1
      -v Directory=QM1
      -v Prefix=/var/mqm
      -v DataPath=/MQHA/qmgrs/QM1
      ```
10. For Server2, complete the following steps:
    a. Mount the exported file system /MQHA by entering the following command: **mount -t** nfs4 **-o** hard,intr *Server1:/ /MQHA*
    b. Paste the queue manager configuration command into Server2:

       ```
        addmqinf -s QueueManager
       -v Name=QM1
       -v Directory=QM1
       -v Prefix=/var/mqm
       -v DataPath=/MQHA/qmgrs/QM1
       ```
11. Start the queue manager instances, in either order, with the **-x** parameter: **strmqm -x** QM1.

    The command used to start the queue manager instances must be issued from the same IBM MQ installation as the **addmqinf** command. To start and stop the queue manager from a different installation, you must first set the installation associated with the queue manager using the **setmqm** command. For more information, see setmqm.

Verifying the multi-instance queue manager on Linux: 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 “Create a multi-instance queue manager on Linux” on page 1278. 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.

13. The '*' allows all machines that can reach this one mount /MQHA for read/write. Restrict access on a production machine.

Verifying failover using IBM MQ Explorer

Before using the sample applications to verify failover, run the IBM MQ Explorer on each server. Add both queue manager instances to each explorer using the **Add Remote Queue Manager > Connect directly to a multi-instance queue manager** wizard. Ensure that both instances are running, permitting standby. Close the window running the VMware image with the active instance, virtually powering off the server, or stop the active instance, allowing switchover to standby instance.

Note: If you power off the server, make sure that it is not the one hosting `/MQHA` !

Note: The **Allow switchover to a standby instance** option might not be available on the Stop Queue Manager dialog. The option is missing because the queue manager is running as a single instance queue manager. You must have started it without the **Permit a standby instance** option. If your request to stop the queue manager is rejected, look at the Details window, it is possibly because there is no standby instance running.

Verifying failover using the sample programs

Choose a server to be to run the active instance

You might have chosen one of the servers to host the `MQHA` directory or file system. If you plan to test failover by closing the VMware window running the active server, make sure that it is not the one hosting `MQHA` !

On the server running the active queue manager instance

Note: Running the `SVRCONN` channel with the `MCAUSER` set to `mqm`, is a convenience to reduce the number of configuration steps in the example. If another user ID is chosen, and your system is set up differently to the one used in the example, you might experience access permission problems. Do not use `mqm` as a `MCAUSER` on an exposed system; it is likely to compromise security greatly.

1. Modify *ipaddr1* and *ipaddr2* and save the following commands in `/MQHA/hasamples.tst.`

```
DEFINE QLOCAL(SOURCE) REPLACE
DEFINE QLOCAL(TARGET) REPLACE
DEFINE CHANNEL(CHANNEL1) CHLTYPE(SVRCONN)  TRPTYPE(TCP) +
MCAUSER('mqm') REPLACE
DEFINE CHANNEL(CHANNEL1) CHLTYPE(CLNTCONN) TRPTYPE(TCP) +
CONNAME(' ipaddr1 (1414), ipaddr2
(1414)') QMNAME(QM1) REPLACE
START  CHANNEL(CHANNEL1)
DEFINE LISTENER(LISTENER.TCP) TRPTYPE(TCP) CONTROL(QMGR)
DISPLAY LISTENER(LISTENER.TCP) CONTROL
START  LISTENER(LISTENER.TCP)
DISPLAY LSSTATUS(LISTENER.TCP) STATUS
```

2. Open a terminal window with the path `/MQHA` and run the command:

```
runmqsc -m QM1 < hasamples.tst
```

3. Verify that the listener is running and has queue manager control, either by inspecting the output of the **`runmqsc`** command.

```
LISTENER(LISTENER.TCP)CONTROL(QMGR)
LISTENER(LISTENER.TCP)STATUS(RUNNING)
```

   Or, using the IBM MQ Explorer that the TCPIP listener is running and has `Control = Queue Manager`.

On the client

1. Copy the client connection table AMQCLCHL.TAB from `/MQHA/qmgrs/QM1.000/@ipcc` on the server to `C:\` on the client.
2. Open a command prompt with the path `C:\` and set the environment variable `MQCHLLIB` to point to the client channel definition table (CCDT)

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

3. At the command prompt type the commands:

   ```
   start amqsghac TARGET QM1
   start amqsmhac -s SOURCE -t TARGET -m QM1
   start amqsphac SOURCE QM1
   ```

On the server running the active queue manager instance

1. Either:
   - Close the window running the VMware image with the active server instance.
   - Using the IBM MQ Explorer, stop the active queue manager instance, allowing switchover to the standby instance and instructing reconnectable clients to reconnect.
2. The three clients eventually detect the connection is broken, and then reconnect. In this configuration, if you close the server window, it is taking about seven minutes for all three connections to be reestablished. Some connections are reestablished well before others.

Results

```
N:\>amqsphac SOURCE QM1
Sample AMQSPHAC start
target queue is SOURCE
message Message 1
message Message 2
message Message 3
message Message 4
message Message 5
17:05:25 : EVENT : Connection Reconnecting (Delay: 0ms)
17:05:47 : EVENT : Connection Reconnecting (Delay: 0ms)
17:05:52 : EVENT : Connection Reconnected
message Message 6
message Message 7
message Message 8
message Message 9
```

```
N:\>amqsmhac -s SOURCE -t TARGET -m QM1
Sample AMQSMHA0 start

17:05:25 : EVENT : Connection Reconnecting (Delay: 97ms)
17:05:48 : EVENT : Connection Reconnecting (Delay: 0ms)
17:05:53 : EVENT : Connection Reconnected
```

```
N:\>amqsghac TARGET QM1
Sample AMQSGHAC start
message Message 1
message Message 2
message Message 3
message Message 4
message Message 5
17:05:25 : EVENT : Connection Reconnecting (Delay: 156ms)
17:05:47 : EVENT : Connection Reconnecting (Delay: 0ms)
17:05:52 : EVENT : Connection Reconnected
message Message 6
message Message 7
message Message 8
message Message 9
```

Deleting a multi-instance queue manager: Multi

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*.

UNIX Linux On UNIX and Linux systems, if you have placed a common `mqs.ini` file in network storage and referenced it from all the servers by setting the `AMQ_MQS_INI_LOCATION` environment variable on each server, then you need to delete the queue manager from only one of its servers as there is only one `mqs.ini` file to update.

Example

First server
  dltmqm QM1

Other servers where instances are defined
  rmvmqinf QM1, or

  dltmqm QM1

Starting and stopping a multi-instance queue manager: Multi

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: Multi

On Mulitplatforms, 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: Multi

A multi-instance queue manager is resilient because it uses a standby queue manager instance to restore queue manager availability after failure.

Replicating queue manager instances is a very effective way to improve the availability of queue manager processes. Using a simple availability model, purely for illustration: if the reliability of one instance of a queue manager is 99% (over one year, cumulative downtime is 3.65 days) then adding another instance of the queue manager increases the availability to 99.99% (over one year, cumulative downtime of about an hour).

This is too simple a model to give you practical numeric estimates of availability. To model availability realistically, you need to collect statistics for the mean time between failures (MTBF) and the mean time to repair (MTTR), and the probability distribution of time between failures and of repair times.

The term, multi-instance queue manager, refers to the combination of active and standby instances of the queue manager that share the queue manager data and logs. Multi-instance queue managers protect you against the failure of queue manager processes by having one instance of the queue manager active on one server, and another instance of the queue manager on standby on another server, ready to take over automatically should the active instance fail.

Failover or switchover: Multi

A standby queue manager instance takes over from the active instance either on request (switchover), or when the active instance fails (failover).

- *Switchover* takes place when a standby instance starts in response to the **endmqm** `-s` command being issued to the active queue manager instance. You can specify the **endmqm** parameters `-c`, `-i` or `-p` to control how abruptly the queue manager is stopped.

  Note: Switchover only takes place if a standby queue manager instance is already started. The **endmqm** `-s` command releases the active queue manager lock and permits switchover: it does not start a standby queue manager instance.
- *Failover* occurs when the lock on queue manager data held by the active instance is released because the instance appeared to stop unexpectedly (that is, without an **endmqm** command being issued).

When the standby instance takes over as the active instance, it writes a message to the queue manager error log.

Reconnectable clients are automatically reconnected when a queue manager fails or switches over. You do not need to include the `-r` flag on the **endmqm** command to request client reconnection. Automatic client reconnect is not supported by IBM MQ classes for Java.

If you find that you cannot restart a failed instance, even though failover has occurred and the standby instance has become active, check to see whether applications connected locally to the failed instance have disconnected from the failed instance.

Locally connected applications must end or disconnect from a failed queue manager instance in order for the failed instance to be restarted. Any locally connected applications using shared bindings (which is the default setting) which hold on to a connection to a failed instance act to prevent the instance from being restarted.

If it is not possible to end the locally connected applications, or ensure that they disconnect when the local queue manager instance fails, consider using isolated bindings. Locally connected applications using isolated bindings do not prevent the local queue manager instance from being restarted, even if they do not disconnect.

Channel and client reconnection: Multi

Channel and client reconnection is an essential part of restoring message processing after a standby queue manager instance has become active.

Multi-instance queue manager instances are installed on servers with different network addresses. You need to configure IBM MQ channels and clients with connection information for all queue manager instances. When a standby takes over, clients and channels are automatically reconnected to the newly active queue manager instance at the new network address. Automatic client reconnect is not supported by IBM MQ classes for Java.

The design is different from the way high availability environments such as HA-CMP work. HA-CMP provides a virtual IP address for the cluster and transfer the address to the active server. IBM MQ reconnection does not change or reroute IP addresses. It works by reconnecting using the network addresses you have defined in channel definitions and client connections. As an administrator, you need to define the network addresses in channel definitions and client connections to all instances of any multi-instance queue manager. The best way to configure network addresses to a multi-instance queue manager depends on the connection:

Queue manager channels
: The `CONNAME` attribute of channels is a comma-separated list of connection names; for example, `CONNAME('127.0.0.1(1234), 192.0.2.0(4321)')`. The connections are tried in the order specified in the connection list until a connection is successfully established. If no connection is successful, the channel attempts to reconnect.

Cluster channels
: Typically, no additional configuration is required to make multi-instance queue managers work in a cluster.

  If a queue manager connects to a repository queue manager, the repository discovers the network address of the queue manager. It refers to the `CONNAME` of the `CLUSRCVR` channel at the queue manager. On TCPIP, the queue manager automatically sets the `CONNAME` if you omit it, or configure it to blanks. When a standby instance takes over, its IP address replaces the IP address of the previous active instance as the `CONNAME`.

  If it is necessary, you can manually configure `CONNAME` with the list of network addresses of the queue manager instances.

Client connections
: Client connections can use connection lists, or queue manager groups to select alternative connections. Clients need to be compiled to run with IBM WebSphere MQ Version 7.0.1 client libraries or better. They must be connected to at least a Version 7.0.1 queue manager.

When failover occurs, reconnection takes some time. The standby queue manager has to complete its startup. The clients that were connected to the failed queue manager have to detect the connection failure, and start a new client connection. If a new client connection selects the standby queue manager that has become newly active, then the client is reconnected to the same queue manager.

If the client is in the middle of an MQI call during the reconnection, it must tolerate an extended wait before the call completes.

If the failure takes place during a batch transfer on a message channel, the batch is rolled back and restarted.

Switching over is faster than failing over, and takes only as long as stopping one instance of the queue manager and starting another. For a queue manager with only few log records to replay, at best switchover might take of the order of a few seconds. To estimate how long failover takes, you need to add the time that it takes for the failure to be detected. At best the detection takes of the order of 10 seconds, and might be several minutes, depending on the network and the file system.

Application recovery: ▶ Multi

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 loosing 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 1288.

You have three alternative design patterns for IBM MQ MQI client applications following failover. Only one of them does not need to detect the failover.

No reconnection

In this pattern, the application stops all processing on the current connection when the connection is broken. For the application to continue processing, it must establish a new connection with the queue manager. The application is entirely responsible for transferring any state information it requires to continue processing on the new connection. Existing client applications that reconnect with a queue manager after losing their connection are written in this way.

The client receives a reason code, such as `MQRC_CONNECTION_BROKEN`, or `MQRC_Q_MGR_NOT_AVAILABLE` from the next MQI call after the connection is lost. The application must discard all its IBM MQ state information, such as queue handles, and issue a new MQCONN or MQCONNX call to establish a new connection, and then reopen the IBM MQ objects it needs to process.

The default MQI behavior is for the queue manager connection handle to become unusable after a connection with the queue manager is lost. The default is equivalent to setting the `MQCNO_RECONNECT_DISABLED` option on MQCONNX to prevent application reconnection after failover.

Failover tolerant
: Write the application so it is unaffected by failover. Sometimes careful error handling is sufficient to deal with failover.

Reconnection aware
: Register an `MQCBT_EVENT_HANDLER` event handler with the queue manager. The event handler is posted with `MQRC_RECONNECTING` when the client starts to try to reconnect to the server, and `MQRC_RECONNECTED` after a successful reconnection. You can then run a routine to reestablish a predictable state so that the client application is able to continue processing.

Recovery of an automatically reconnected client

Failover is an unexpected event, and for an automatically reconnected client to work as designed the consequences of reconnection must be predictable.

A major element of turning an unexpected failure into a predictable and reliable recovery is the use of transactions.

In the previous section, an example, 2 on page 1287, was given of an IBM MQ MQI client using a local transaction to coordinate MQGET and MQPUT. The client issues an MQCMIT or MQBACK call in response to a MQRC_BACKED_OUT error and then resubmits the backed out transaction. The queue manager failure causes the transaction to be backed out, and the behavior of the client application ensures no transactions, and no messages, are lost.

Not all program state is managed as part of a transaction, and therefore the consequences of reconnection become harder to understand. You need to know how reconnection changes the state of an IBM MQ MQI client in order to design your client application to survive queue manager failover.

You might decide to design your application without any special failover code, handling reconnection errors with the same logic as other errors. Alternatively, you might choose to recognize that reconnection requires special error processing, and register an event handler with IBM MQ to run a routine to handle failover. The routine might handle the reconnection processing itself, or set a flag to indicate to the main program thread that when it resumes processing it needs to perform recovery processing.

The IBM MQ MQI client environment is aware of failover itself, and restores as much context as it can, following reconnection, by storing some state information in the client, and issuing additional MQI calls on behalf of the client application to restore its IBM MQ state. For example, handles to objects that were open at the point of failure are restored, and temporary dynamic queues are opened with the same name. But there are changes that are unavoidable and you need your design to deal with these changes. The changes can be categorized into five kinds:

1. New, or previously undiagnosed errors, are returned from MQI calls until a consistent new context state is restored by the application program.

   An example of receiving a new error is the return code `MQRC_CONTEXT_NOT_AVAILABLE` when trying to pass context after saving context before the reconnection. The context cannot be restored after reconnection because the security context is not passed to an unauthorized client program. To do so would let a malicious application program obtain the security context.

   Typically, applications handle common and predictable errors in a carefully designed way, and relegate uncommon errors to a generic error handler. The error handler might disconnect from IBM MQ and reconnect again, or even stop the program altogether. To improve continuity, you might need to deal with some errors in a different way.
2. Non-persistent messages might be lost.
3. Transactions are rolled back.
4. MQGET or MQPUT calls used outside a sync point might be interrupted with the possible loss of a message.
5. Timing induced errors, due to a prolonged wait in an MQI call.

Some details about lost context are listed in the following section.

- Non-persistent messages are discarded, unless put to a queue with the `NPMCLASS(HIGH)` option, and the queue manager failure did not interrupt the option of storing non-persistent messages on shutdown.
- A non-durable subscription is lost when a connection is broken. On reconnection, it is re-established. Consider using a durable subscription.
- The get-wait interval is recomputed; if its limit is exceeded it returns `MQRC_NO_MSG_AVAILABLE`. Similarly, subscription expiry is recomputed to give the same overall expiry time.
- The position of the browse cursor in a queue is lost; it is typically reestablished before the first message.
  - MQGET calls that specify `MQGMO_BROWSE_MSG_UNDER_CURSOR` or `MQGMO_MSG_UNDER_CURSOR`, fail with reason code `MQRC_NO_MSG_AVAILABLE`.
  - Messages locked for browsing are unlocked.
  - Browse marked messages with handle scope are unmarked and can be browsed again.
  - Cooperatively browse marked messages are unmarked in most cases.
- Security context is lost. Attempts to use saved message context, such as putting a message with `MQPMO_PASS_ALL_CONTEXT` fail with `MQRC_CONTEXT_NOT_AVAILABLE`.
- Message tokens are lost. MQGET using a message token returns the reason code `MQRC_NO_MSG_AVAILABLE`.

  Note: *MsgId* and *CorrelId*, as they are part of the message, are preserved with the message during failover, and so MQGET using `MsgId` or `CorrelId` work as expected.
- Messages put on a queue under sync point in an uncommitted transaction are no longer available.
- Processing messages in a logical order, or in a message group, results in a return code of `MQRC_RECONNECT_INCOMPATIBLE` after reconnection.
- An MQI call might return `MQRC_RECONNECT_FAILED` rather than the more general `MQRC_CONNECTION_BROKEN` that clients typically receive today.
- Reconnection during an MQPUT call outside sync point returns `MQRC_CALL_INTERRUPTED` if the IBM MQ MQI client does not know if the message was delivered to the queue manager successfully. Reconnection during MQCMIT behaves similarly.
- `MQRC_CALL_INTERRUPTED` is returned - after a successful reconnect - if the IBM MQ MQI client has received no response from the queue manager to indicate the success or failure of
  - the delivery of a persistent message using an MQPUT call outside sync point.
  - the delivery of a persistent message or a message with default persistence using an MQPUT1 call outside sync point.
  - the commit of a transaction using an MQCMIT call. The response is only ever returned after a successful reconnect.
- Channels are restarted as new instances (they might also be different channels), and so no channel exit state is retained.
- Temporary dynamic queues are restored as part of the process of recovering reconnectable clients that had temporary dynamic queues open. No messages on a temporary dynamic queue are restored, but applications that had the queue open, or had remembered the name of the queue, are able to continue processing.

  There is the possibility that if the queue is being used by an application other than the one that created it, that it might not be restored quickly enough to be present when it is next referenced. For example, if a client creates a temporary dynamic queue as a reply-to queue, and a reply message is to be placed on the queue by a channel, the queue might not be recovered in time. In this case, the channel would typically place the reply-to message on the dead letter queue.

  If a reconnectable client application opens a temporary dynamic queue by name (because another application has already created it), then when reconnection occurs, the IBM MQ MQI client is unable to 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: Multi

High availability solutions using multi-instance queue managers must include a mechanism to recover data after a storage failure.

A multi-instance queue manager increases the availability of queue manager processes, but not the availability of other components, such as the file system, that the queue manager uses to store messages, and other information.

One way to make data highly available is to use networked resilient data storage. You can either build your own solution using a networked file system and resilient data storage, or you can buy an integrated solution. If you want to combine resilience with disaster recovery, then asynchronous disk replication, which permits disk replication over tens, or hundreds of kilometers, is available.

You can configure the way different IBM MQ directories are mapped to storage media, to make the best use of the media. For *multi-instance* queue managers there is an important distinction between two types of IBM MQ directories and files.

Directories that must be shared between the instances of a queue manager.
The information that must be shared between different instances of a queue manager is in two directories: the `qmgrs` and `logs` directories. The directories must be on a shared networked file system. You are advised to use a storage media that provides continuous high availability and excellent performance because the data is constantly changing as messages are created and deleted.

Directories and files that do not *have* to be shared between instances of a queue manager.
Some other directories do not have to be shared between different instances of a queue manager, and are quickly restored by means other than using a mirrored file system.

- IBM MQ executable files and the tools directory. Replace by reinstalling or by backing up and restoring from a backed up file archive.
- Configuration information that is modified for the installation as a whole. The configuration information is either managed by IBM MQ, such as the `mqs.ini` file on Windows, UNIX and Linux systems, or part of your own configuration management such as **`MQSC`** configuration scripts. Back up and restore using a file archive.
- Installation-wide output such as traces, error logs and FFDC files. The files are stored in the `errors` and `trace` subdirectories in the default data directory. The default data directory on UNIX and Linux systems is `/var/mqm`. On Windows the default data directory is the IBM MQ installation directory.

You can also use a backup queue manager to take regular media backups of a multi-instance queue manager using linear logging. A backup queue manager does not provide recovery that is as fast as from a mirrored file system, and it does not recover changes since the last backup. The backup queue manager mechanism is more appropriate for use in off-site disaster recovery scenarios than recovering a queue manager after a localized storage failure.

Combining IBM MQ Availability solutions

Applications are using other IBM MQ capabilities to improve availability. Multi-instance queue managers complement other high availability capabilities.

IBM MQ Clusters increase queue availability

You can increase queue availability by creating multiple definitions of a cluster queue; up to one of every queue on each manager in the cluster.

Suppose a member of the cluster fails and then a new message is sent to a cluster queue. Unless the message *has* to go to the queue manager that has failed, the message is sent to another running queue manager in the cluster that has a definition of the queue.

Although clusters greatly increase availability, there are two related failure scenarios that result in messages getting delayed. Building a cluster with multi-instance queue managers reduces the chance of a message being delayed.

Marooned messages
: If a queue manager in the cluster fails, no more messages that can be routed to other queue managers in the cluster are routed to the failed queue manager. Messages that have already been sent are marooned until the failed queue manager is restarted.

Affinities
: Affinity is the term used to describe information shared between two otherwise separate computations. For example, an affinity exists between an application sending a request message to a server and the same application expecting to process the reply. Another example would be a sequence of messages, the processing of each message depending on the previous messages.

  If you send messages to clustered queues you need to consider affinities. Do you need to send successive messages to the same queue manager, or can each message go to any member of the cluster?

  If you do need to send messages to the same queue manager in the cluster and it fails, the messages wait in the transmission queue of the sender until the failed cluster queue manager is running again.

If the cluster is configured with multi-instance queue managers the delay waiting for the failed queue manager to restart is limited to the order of a minute or so while the standby takes over. When the standby is running, marooned messages resume processing, channels to the newly activated queue manager instance are started, and the messages that were waiting in transmission queues start flowing.

A possible way to configure a cluster to overcome messages being delayed by a failed queue manager, is to deploy two different queue managers to each server in the cluster, and arrange for one to be the active and one to be the standby instance of the different queue managers. This is an active-standby configuration, and it increases the availability of the cluster.

As well as having the benefits of reduced administration and increased scalability, clusters continue to provide additional elements of availability to complement multi-instance queue managers. Clusters protect against other types of failure that affect both the active and standby instances of a queue manager.

Uninterrupted service
: A cluster provides an uninterrupted service. New messages received by the cluster are sent to active queue managers to be processed. Do not rely on a multi-instance queue manager to provide an uninterrupted service because it takes time for the standby queue manager to detect the failure and complete its startup, for its channels to be reconnected, and for failed batches of messages to be resubmitted.

Localized outage

There are practical limitations to how far apart the active, standby, and file system servers can be, as they need to interact at millisecond speeds to deliver acceptable performance.

Clustered queue managers require interaction speeds of the order of many seconds, and can be geographically dispersed anywhere in the world.

Operational error

By using two different mechanisms to increase availability you reduce the chances that an operational error, such as a human error, compromises your availability efforts.

z/OS

Queue-sharing groups increase message processing availability

Queue-sharing groups, provided only on z/OS, allow a group of queue managers to share servicing a queue. If one queue manager fails, the other queue managers continue to process all the messages on the queue. Multi-instance queue managers are not supported on z/OS and complement queue-sharing groups only as part of a wider messaging architecture.

IBM MQ Clients increase application availability

IBM MQ MQI client programs can connect to different queue managers in a queue manager group based on queue manager availability, connection weightings, and affinities. By running an application on a different machine from the one on which the queue manager is running, you can to improve the overall availability of a solution as long as there is a way to reconnect the application if the queue manager instance it is connected to fails.

Queue manager groups are used to increase client availability by uncoupling a client from a queue manager that is stopped, and load balancing client connections across a group of queue managers, rather like an IP sprayer. The client application must have no affinities with the failed queue manager, such as a dependency on a particular queue, or it cannot resume processing.

Automatic client reconnection and multi-instance queue managers increase client availability by resolving some affinity problems. Automatic client reconnect is not supported by IBM MQ classes for Java.

You can set the `MQCNO` option `MQCNO_RECONNECT_Q_MGR`, to force a client to reconnect to the same queue manager:

1. If the previously connected single instance queue manager is not running the connection is tried again until the queue manager is running again.
2. If the queue manager is configured as a multi-instance queue manager, then the client reconnects to whichever instance is active.

By automatically reconnecting to the same queue manager, much of the state information the queue manager was holding on behalf of the client, such as the queues it had open and the topic it was subscribed to, are restored. If the client had opened a dynamic reply-to queue to receive a reply to a request, the connection to the reply-to queue is restored too.

RDQM high availability

Linux V 9.0.4

RDQM (replicated data queue manager) is a high availability solution that is available on Linux 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 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: The IBM MQ Console does not support replicated data queue managers.

The group configuration of the three nodes is handled by Pacemaker. The replication between the three nodes is handled by DRBD.

The following figure shows a typical deployment with an RDQM running on each of the three nodes in the HA group.

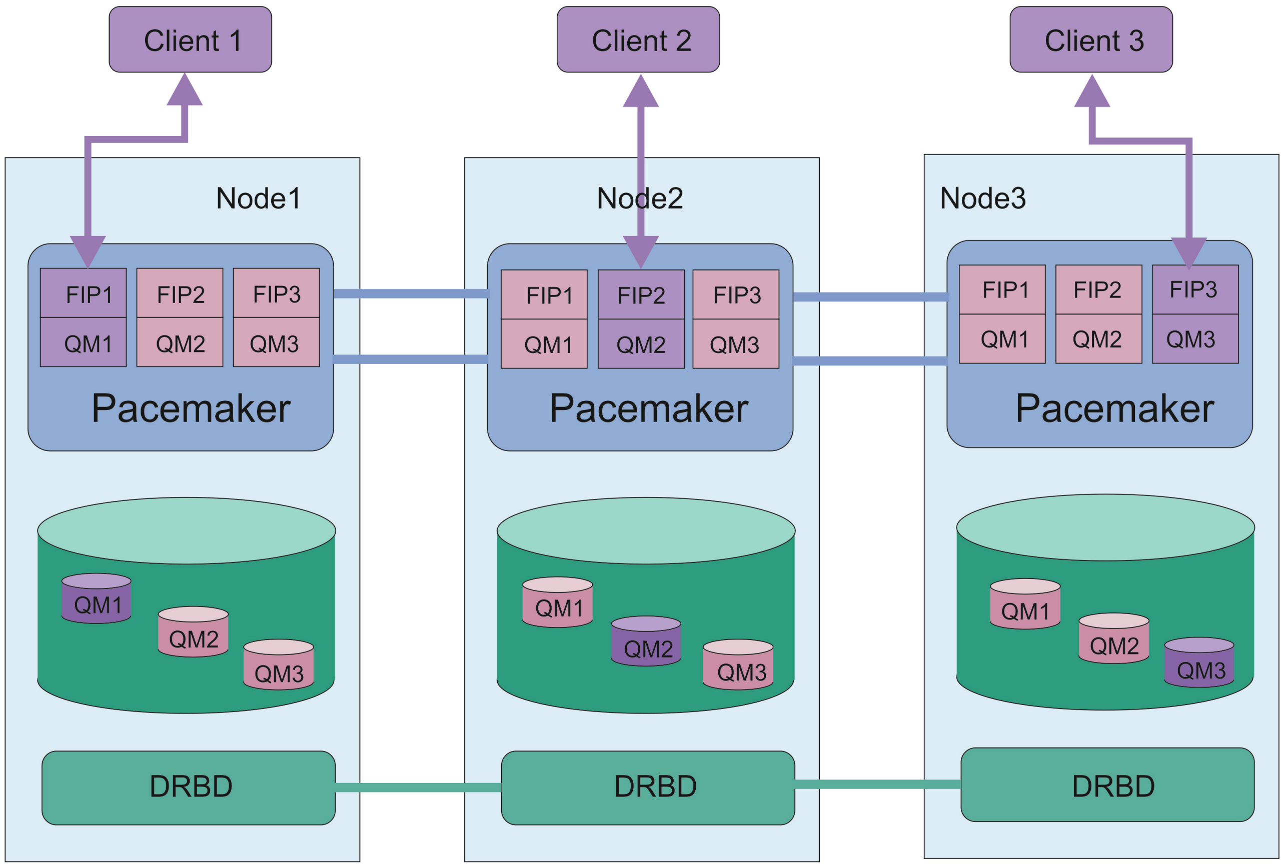


Figure 158. Example of HA group with three RDQMs

In the next figure, Node3 has failed, the Pacemaker links have been lost, and queue manager QM3 runs on Node2 instead.

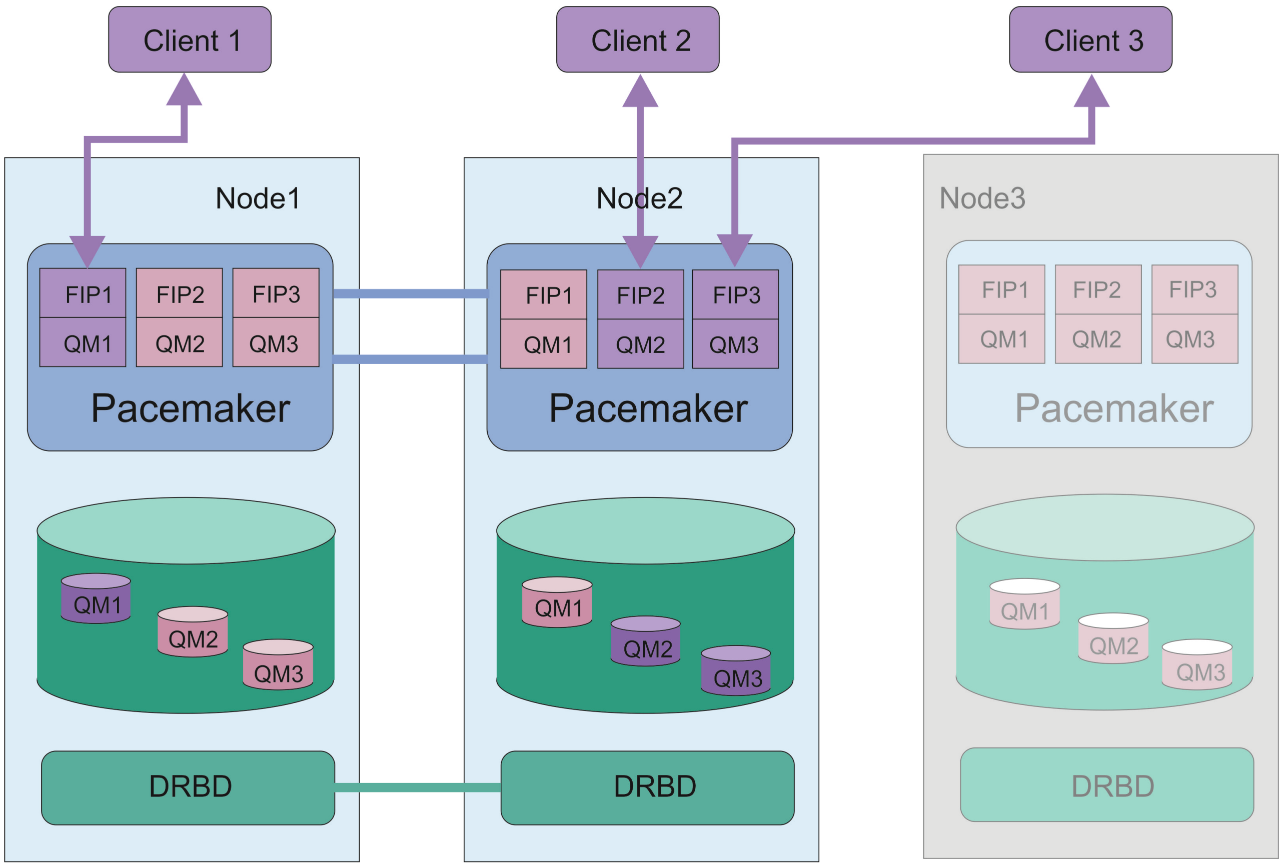


Figure 159. Example after node3 fails

Requirements for RDQM HA solution: Linux V 9.0.4

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 `drbdpool`. 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).
- 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.
- 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 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 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 an mqm user 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.
- If you configure password-less SSH for your mqm user, that user must have the same UID 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/rdqmadm
/opt/mqm/bin/rdqmstatus
```

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.

Setting up passwordless SSH: Linux V 9.0.4

You can set up passwordless SSH so that you only need issue configuration commands on one node in the HA group.

Before you begin

You must set up sudo privileges for the mqm user.

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 .

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.

Procedure

1. On each of the three nodes, complete the following steps to set up the mqm user and generate an SSH key:
   a. Change the mqm home directory to /home/mqm:
      ```
      usermod -d /home/mqm mqm
      ```
   b. Create the /home/mqm directory:
      ```
      mkhomedir_helper mqm
      ```
   c. Add the mqm password:
      ```
      passwd mqm
      ```
   d. Run the interactive shell as mqm:
      ```
      su mqm
      ```
   e. Generate the mqm authentication key:
      ```
      ssh-keygen -t rsa -f /home/mqm/.ssh/id_rsa -N ''
      ```
2. 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:
   a. 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
      ```
   b. 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.
   c. Exit the interactive shell as mqm:
      ```
      exit
      ```
3. On each node, as root, complete the following steps to remove the mqm password and lock the id:

a. Remove the mqm password:

```
passwd -d mqm
```

b. Lock mqm:

```
passwd -l mqm
```

Defining the Pacemaker cluster (HA group): ▶ Linux ▶ V 9.0.4

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

You can create the Pacemaker cluster as a user in the mqm group if the 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 for all of the nodes in the Pacemaker cluster. You can specify that the Pacemaker cluster uses one, two, or three IP addresses. The interface that is used for synchronous data replication is named the 'replication interface'. The interface must have sufficient bandwidth to support replication requirements given the expected workload of all the RDQMs running in the HA Group. The primary and secondary interfaces are used for the Pacemaker to monitor the system, but Pacemaker can use the replication interface for this purpose, if required.

The following example file shows the configuration for an example Pacemaker cluster that uses a separate IP address for each interface:

```
Node:
  HA_Primary=192.168.4.1
  HA_Alternate=192.168.5.1
  HA_Replication=192.168.6.1
Node:
  HA_Primary=192.168.4.2
  HA_Alternate=192.168.5.2
  HA_Replication=192.168.6.2
Node:
  HA_Primary=192.168.4.3
  HA_Alternate=192.168.5.3
  HA_Replication=192.168.6.3
```

The following example file shows the configuration for an example Pacemaker cluster that uses the same IP address for each interface. In this case you only specify the Replication interface:

```
Node:
  HA_Replication=192.168.4.1
Node:
  HA_Replication=192.168.4.2
Node:
  HA_Replication=192.168.4.3
```

If you wanted to use two IP addresses, your rdqm.ini file has a Primary and a Replication field for each node, but no Alternate field:

```
Node:
  HA_Primary=192.168.4.1
  HA_Replication=192.168.5.1
Node:
  HA_Primary=192.168.4.2
  HA_Replication=192.168.5.2
Node:
  HA_Primary=192.168.4.3
  HA_Replication=192.168.5.3
```

Procedure

- To define the Pacemaker cluster as an mqm user:
  1. Ensure that the user mqm can use **sudo** to run commands and can optionally connect to each server using SSH without a password.
  2. Edit the /var/mqm/rdqm.ini file on one of the three servers so that the file defines the Pacemaker cluster.
  3. Run the following command:
     ```
     rdqmadm -c
     ```
     (If you cannot SSH without a password, you must copy the .ini file to each server and run the command on each server.)
- To define the Pacemaker cluster as user root:
  1. Edit the /var/mqm/rdqm.ini file on one of the three servers so that the file defines the cluster.
  2. Copy the file to the other two servers that will be nodes in the Pacemaker cluster.
  3. Run the following command as root on each of the three servers:
     ```
     rdqmadm -c
     ```

Related information:
rdqmadm (administer replicated data queue manager cluster)

Deleting the Pacemaker cluster (HA group): Linux V 9.0.4

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 information:
rdqmadm (administer replicated data queue manager cluster)

Creating an HA RDQM: Linux V 9.0.4

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 user can use sudo. If the user can also 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 be root to create an RDQM, and you must run commands on all three nodes.

Procedure

- To create an RDQM as a user in the mqm group:
  1. Ensure that the mqm user can use **sudo** to run commands and can connect to each server using SSH without a password.

2. Enter the following command:

   ```
   crtmqm -sx [-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).

   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`:

  1. Enter the following command on each of the nodes that are to host secondary instances of the RDQM:

     ```
     crtmqm -sxs [-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 command creates a secondary instance of the RDQM.

  2. On the remaining node, enter the following command:

     ```
     crtmqm -sx [-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 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.

Related information:

crtmqm

Deleting an HA RDQM: Linux V 9.0.4

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 information:
dltmqm

Setting the Preferred Location for an RDQM: Linux V 9.0.4

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 information:
rdqmadm (administer replicated data queue manager cluster)

Creating and deleting a floating IP address: Linux V 9.0.4

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.

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.

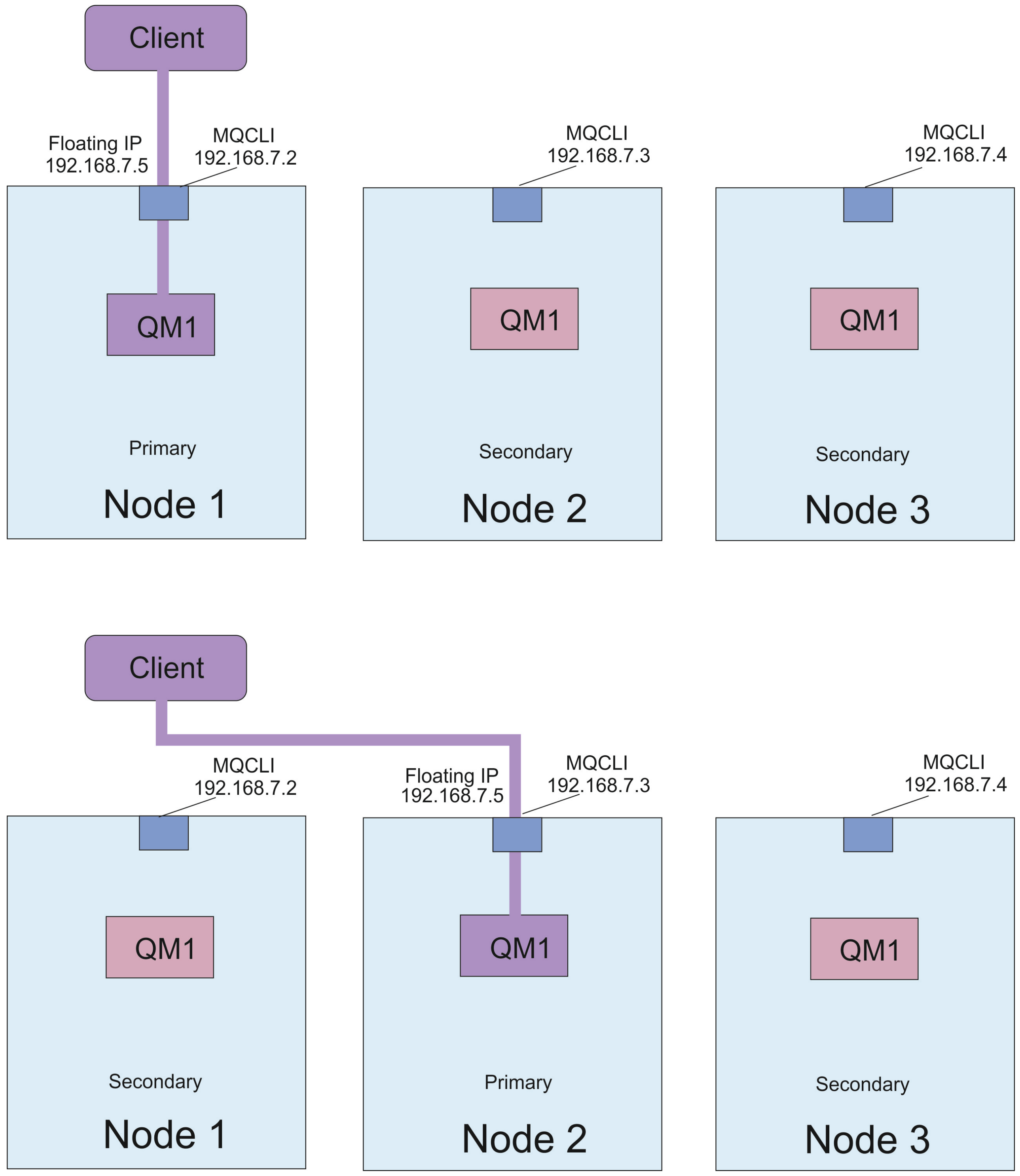


Figure 160. Floating IP address

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.

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.

  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 MQCLI
  ```
- To delete an existing floating IP address, enter the following command:

  ```
  rdqmint -m qmname -d
  ```

Related information:

rdqmint (add or delete floating IP address for RDQM)

Starting, stopping, and displaying the state of an HA RDQM: Linux V 9.0.4

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 (that is, the node on which the queue manager is currently running).

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:

  ```
  dspmq
  ```

  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 information:

dspmq (display queue managers)

endmqm (end queue manager)

strmqm (start queue manager)

Viewing RDQM and HA group status: Linux V 9.0.4

You can view the status of the HA group and of individual 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.

You must be a user in the `mqm` and `haclients` group to run the **rdqmstatus** command. You can run the command on any of the three nodes.

Procedure

- To view the status of a node and the RDQMs that are part of the HA configuration:

  ```
  rdqmstatus
  ```

  The identify of the node that you ran the command on and the status of the RDQMs in the HA configuration is displayed, for example:

  ```
  Node:                                      mqhavm07.exampleco.com

  Queue manager name:                        RDQM8
  Queue manager status:                      Running elsewhere
  HA current location:                       mqhavm08.exampleco.com

  Queue manager name:                        RDQM9
  Queue manager status:                      Running elsewhere
  HA current location:                       mqhavm09.exampleco.com

  Queue manager name:                        RDQM7
  Queue manager status:                      Running
  HA current location:                       This node
  ```

- 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(mqhavm04.example.com) is online
Node mqha05(mqhavm05.example.com) is offline
Node mqha06(mqhavm06.example.com) is online
```

- 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.

  The following table summarizes the information about the current node that can be returned by the **rdqmstatus** command for an RDQM.

Table 128. Current node status

Status attribute	Possible values	When displayed
Node name	*nodename*	Always displayed
Queue manager status	Running Running elsewhere Ended Unavailable	Always displayed
CPU	*n.nn*%	Only shown when current node has primary role (that is, the RDQM is running on this node)
Memory	*nnn*MB used, *y.y*GB allocated	Only shown when current node has primary role (that is, the RDQM is running on this node)
Queue manager file system	*nnn*MB used, *y.y*GB allocated [z%]	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 *status of remote nodes*	All nodes in standby Current node in standby Both remote nodes in standby Different status for each remote node (see next tab Same status for both remote nodes (see next table
HA control	Enabled Disabled Unknown	Always displayed. Shows whether RDQM is under Pacemaker control
HA preferred location	None This node Unknown *nodename*	Always displayed
HA floating IP interface	*Interface_name*	Always displayed
HA floating IP address	*IPV4_address*	Always displayed

The following table summarizes the information that is returned by the **rdqmstatus** command for the other nodes in the HA group.

Table 129. Other node status

Status attribute	Possible values	When displayed
Node name	*nodename*	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 synchronization in progress	*n.n*%	Displayed when synchronization in progress, and command run as `root`
HA estimated synchronization time	*yyyy-mm-dd hh:mm:ss.nnn*	Displayed when synchronization in progress
HA out of sync data	*n*KB	Displayed when remote node unavailable or inconsistent

Example

Example of normal status on primary node:

```
Node:                                   mqhavm07.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 floating IP interface:                Eth4
HA floating IP address:                  192.0.2.4


Node:                                    mqhavm08.exampleco.com
HA status:                               Normal


Node:                                    mqhavm09.exampleco.com
HA status:                               Normal
```

Example of normal status on a secondary node:

```
Node:                                    mqhavm08.exampleco.com
Queue manager status:                    Running elsewhere
HA role:                                 Secondary
HA status:                               Normal
HA control:                              Enabled
HA current location:                     mqhavm07.exampleco.com
HA preferred location:                   mqhavm07.exampleco.com
HA floating IP interface:                Eth4
HA floating IP address:                  192.0.2.4

Node:                                    mqhavm07.exampleco.com
HA status:                               Normal

Node:                                    mqhavm09.exampleco.com
HA status:                               Normal
```

Example of status on primary node when synchronization is in progress:

```
Node:                              mqhavm07.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 floating IP interface:          Eth4
HA floating IP address:            192.0.2.4

Node:                              mqhavm08.exampleco.com
HA status:                         Synchronization in progress
HA synchronization progress:       11.0%
HA estimated time to completion:   2017-09-06 14:55:05

Node:                              mqhavm09.exampleco.com
HA status:                         Synchronization in progress
HA synchronization progress:       11.0%
HA estimated time to completion:   2017-09-06 14:55:06
```

Example of a primary node showing multiple states:

```
Node:                              mqhavm07.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 floating IP interface:          Eth4
HA floating IP address:            192.0.2.4

Node:                              mqhavm08.exampleco.com
HA status:                         Normal

Node:                              mqhavm09.exampleco.com
HA status:                         Inconsistent
```

Related information:

Linux rdqmstatus

Replacing a failed node: Linux V 9.0.4

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.

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:

    1. 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 1298).
    2. Run the `crtmqm -sxs` *`qmanager`* command to recreate each replicated data queue manager (see “Creating an HA RDQM” on page 1299).
- If the replacement node has a different configuration to the failed node:
    1. Delete the replicated data queue managers from the other nodes in the HA group by using the **`dltmqm`** command (see “Deleting an HA RDQM” on page 1300).
    2. Unconfigure the Pacemaker cluster by using the **`rdqmadm -u`** command (see “Deleting the Pacemaker cluster (HA group)” on page 1299).
    3. 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 1298).
    4. Run the `crtmqm -sxs` *`qmanager`* command to recreate each replicated data queue manager (see “Creating an HA RDQM” on page 1299).

Resolving a split-brain situation: V 9.0.4

There are situations where certain failure sequences in an HA group could lead to a split-brain situation being reported.

For example, say all three nodes lose connectivity. If both secondary nodes regain connectivity before the primary node, they form a new quorum and one of them runs the queue manager. When the original primary node regains connectivity, it is possible that a split-brain situation is reported.

In this situation, running `rdqmstatus -m` *`QMname`* on the original primary node shows the HA status as `Inconsistent`:

```
Node:                              node1
Queue manager status:              Running elsewhere
HA role:                           Secondary
HA status:                         Inconsistent
HA control:                        Enabled
HA current location:               hanode2
HA preferred location:             This node
HA floating IP interface:          None
HA floating IP address:            None

Node:                              node2
HA status:                         Inconsistent
HA out of sync data:               8KB

Node:                              node3
HA status:                         Inconsistent
HA out of sync data:               8KB
```

In this instance, you should retain the data on the original secondary nodes (that formed the new quorum). Complete the following steps:

1. On the original primary node, as root, run the following command:
   ```
   drbdadm connect --discard-my-data QMname
   ```
2. On each of the secondary nodes, as root, run the following command:
   ```
   drbdadm connect QMname:first-node-name
   ```

RDQM disaster recovery

▶ Linux ▶ V 9.0.5

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, and a queue manager cannot be configured with both RDQM disaster recovery and RDQM high availability.

You can have several pairs of RDQM queue managers running on a number of different servers. For example, you could have six primary DR queue managers running on the same node, while their secondaries are configured on six different nodes in six different data centers. Equally you could have primary disaster recovery queue managers running on different nodes, while all their secondary disaster recovery queue manages run on the same node. Some example configurations are illustrated in the following diagrams.

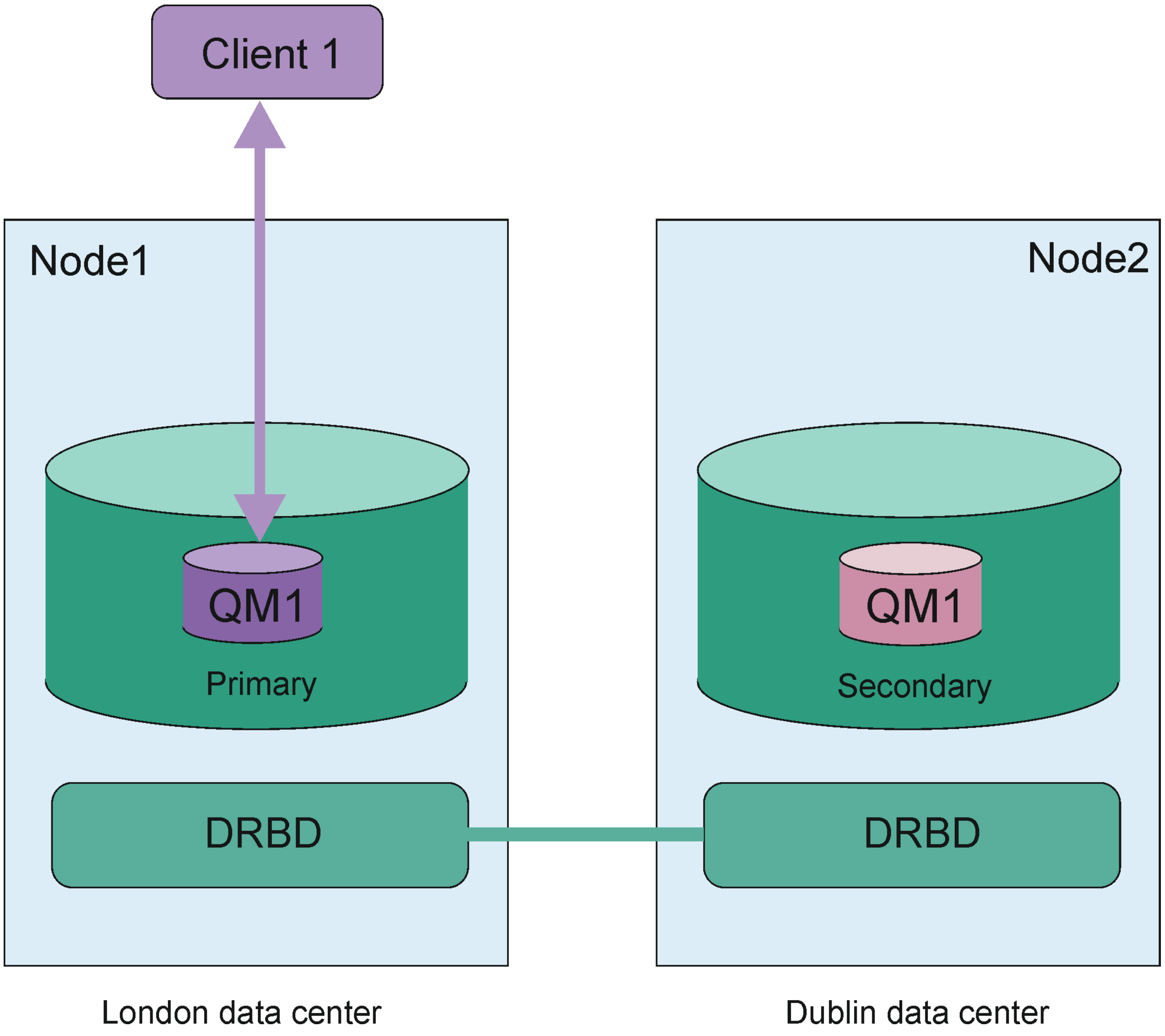


Figure 161. Single RDQM pair

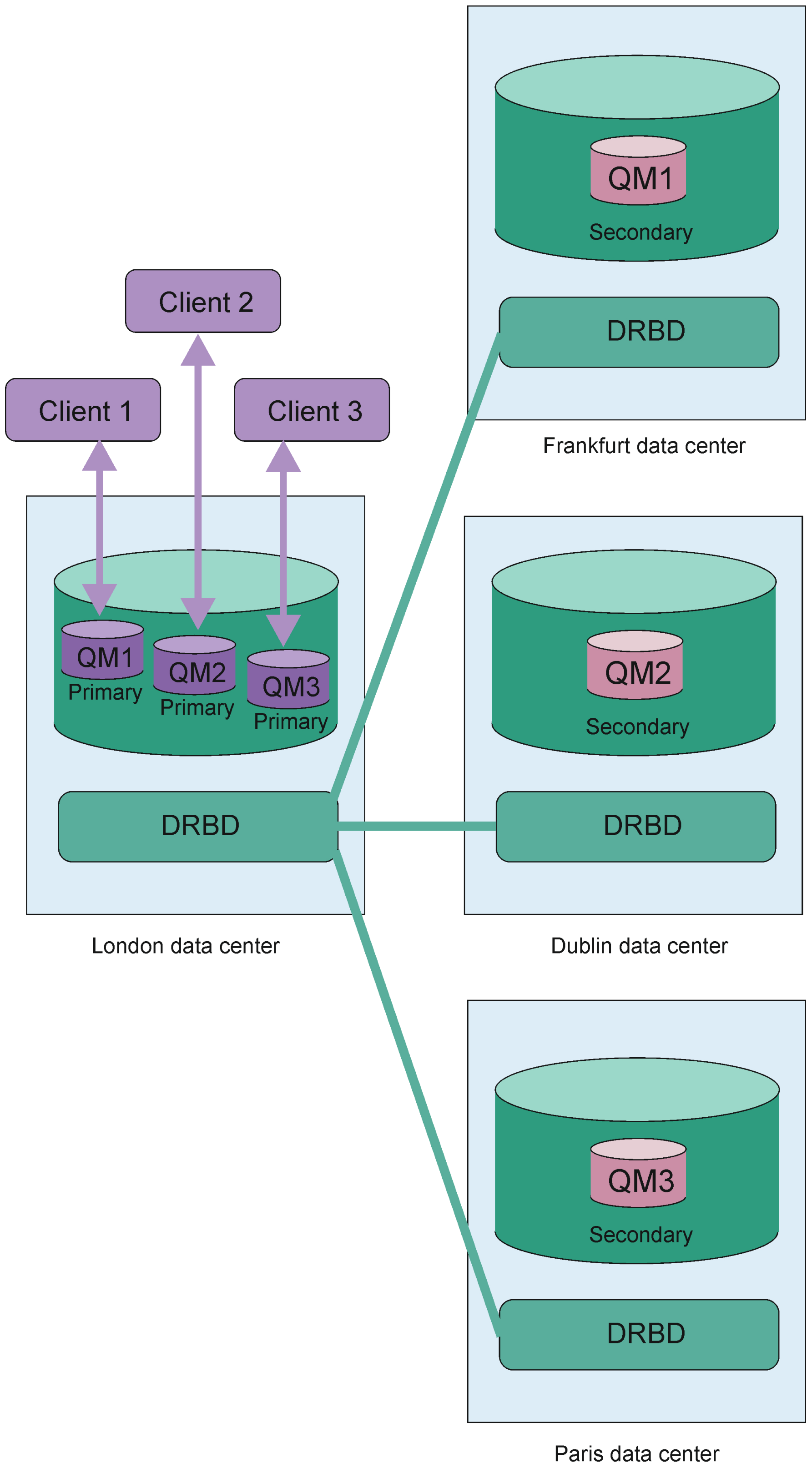


Figure 162. Primary queue managers in same node

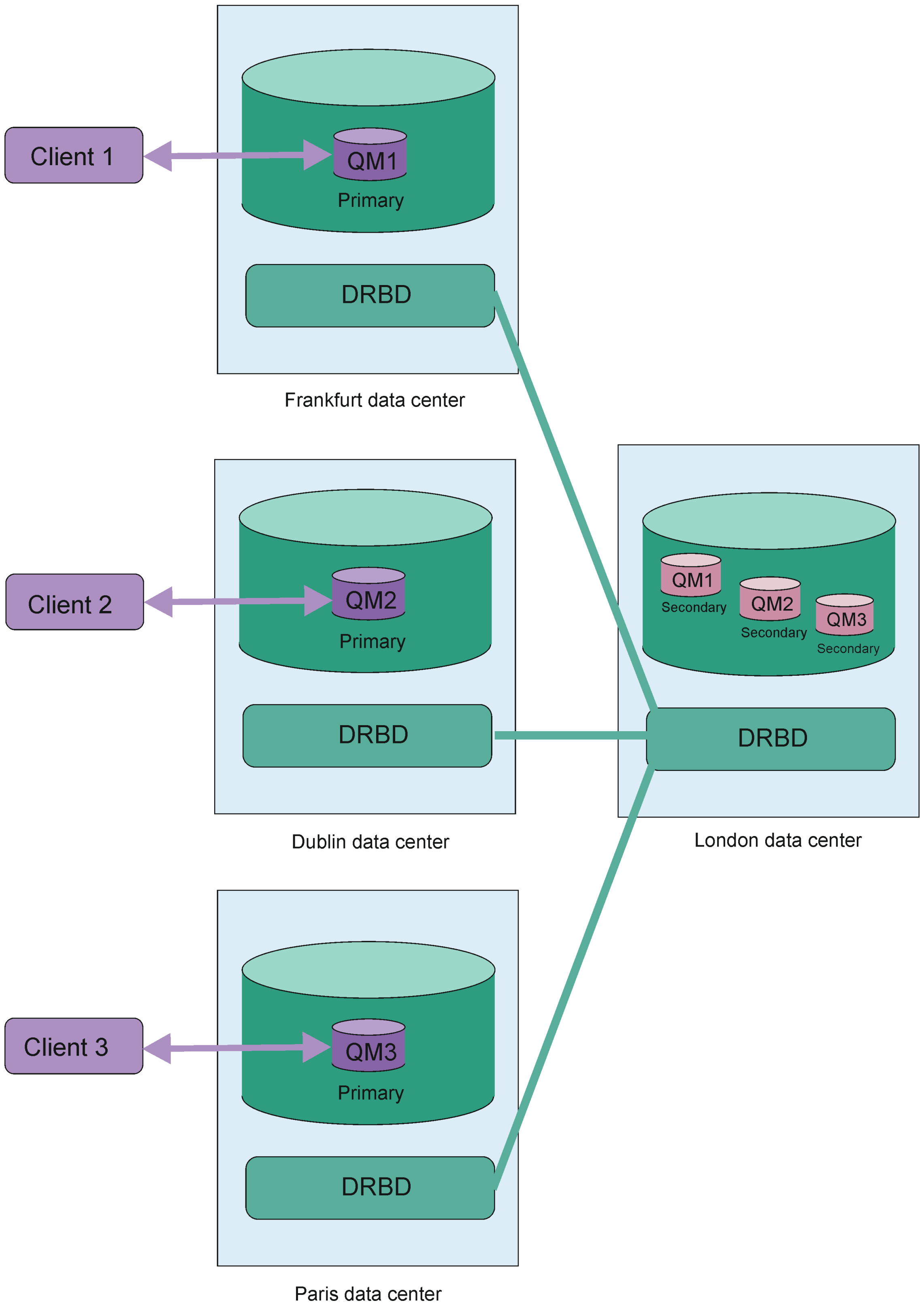


Figure 163. Secondary queue managers in same node

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.

Requirements for RDQM DR solution: Linux V 9.0.5

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 `drbdpool`. 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).
- 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.
- 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 50 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.

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`**
- **`rdqmdr`**

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`**

Creating a disaster recovery RDQM: Linux V 9.0.5

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.

Procedure
- To create a primary DR RDQM:
  1. Enter the following command:
     ```
     crtmqm -rr p [-rt (a | s)] -rl Local_IP -ri Recovery_IP -rn Recovery_Name -rp Port [other_crtmqm_options] [-fs size]
     ```
     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.

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*

-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.

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 tp input on the secondary node to create the secondary instance of the queue manager. You can also use the **`rdqmdr`** 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 1317.

- To create a secondary DR RDQM:
  1. Enter the following command on the node that is to host secondary instances of the RDQM:

```
crtmqm  -rr s [-rt (a | s)] -rl Local_IP -ri Primary_IP -rn Primary_Name  -rp Port [other_crtmqm_options] [-fs siz
```

Where:

-rr s
: Specifies that you are creating the secondary 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.

-rl ***Local_IP***
: Specifies the local IP address to be used for DR replication of this queue manager.

-ri *Primary_IP*
Specifies the IP address of the interface used for replication on the server hosting the primary instance of the queue manager.

-rn *Primary_Name*
Specifies the name of the system that is hosting the primary instance of the queue manager. The name is that value that is returned if you run `uname -n` on that server.

-rp *Port*
Specifies the port to use for DR replication.

other_crtmqm_options
You can optionally specify one or more of these general **crtmqm** options:

– -z

-fs *size*
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. If you have specified a non-default size when creating the primary queue manager, you must specify the same value here.

QMname
Specifies the name of the replicated data queue manager. This must be the same as the name you specified for the primary instance of the queue manager. Note that the name is case sensitive.

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 1319. If they are not displaying this status, delete the secondary instance and recreate it, taking care to use the correct arguments.

Related information:
crtmqm

Deleting a DR RDQM: Linux V 9.0.5

You use the **dltmqm** 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 information:
dltmqm

Managing primary and secondary characteristics of DR RDQMs: Linux V 9.0.5

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 RD RDQM.

Starting, stopping, and displaying the state of a DR RDQM: ▶ Linux ▶ V 9.0.5

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 information:
dspmq
endmqm
strmqm

Viewing DR RDQM status: Linux V 9.0.5

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.

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 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:

  ```
  Queue manager name:                DRQM8
  Queue manager status:              Ended immediately
  DR role:                           Secondary

  Queue manager name:                DRQM7
  Queue manager status:              Running
  DR role:                           Primary
  ```
- To view the status of a particular RDQM, enter the following command:

  ```
  rdqmstatus -m qmname
  ```

  The following table summarizes the information that is returned.

Table 130. Status attributes

Status Attribute	Possible Values	When Displayed
Queue manager status	state (as displayed by dspmq)	Always displayed
CPU	*n.nn*%	Only shown when RDQM on current node has primary role
Memory	*nnn*MB	Only shown when RDQM on current node has primary role
Queue manager file system	*nnn*MB used,*n.n*GB allocated [*n*%]	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
	Partitioned	The queue manager has been started on both nodes while the DR replication network is unavailable

Table 130. Status attributes (continued)

Status Attribute	**Possible Values**	**When Displayed**
	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	*port_number* (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 out of sync data	*n*KB	Displayed when remote node unavailable or inconsistent
DR synchronization progress	*n*%	Displayed when synchronization is in progress
DR estimated time to completion	*YYYY-MM-DD HH:MM:SS*	Displayed hen synchronization is in progress
Snapshot reversion progress	*n*%	Displayed when DR status is `Reverting to snapshot`. The status counts down, so 0% shows completion

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
```

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
```

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 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:                            Mixed
DR type:                              Synchronous
DR port:                              3000
DR local IP address:                  192.168.20.1
DR remote IP address:                 192.168.20.2
```

Related information:

Linux rdqmstatus

Operating in a disaster recovery environment: Linux V 9.0.5

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” on page 1322. For steps to take following permanent failure, see “Replacing a failed node in a disaster recovery configuration” on page 1322.

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” on page 1322.

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.” You must check that the revert to snapshot has completed (by using the **rdqmstatus** command) before you start the queue manager.

Switching over to a recovery node: Linux V 9.0.5

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 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 can also use this procedure to test your recovery node.

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. If you are using this procedure to test your secondary queue manager (that is, the primary instance is still running), you must stop the primary instance and redesignate 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 appliance. 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.

Related information:

strmqm

rdqmdr

Replacing a failed node in a disaster recovery configuration: Linux V 9.0.5

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 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 information:

strmqm

rdqmdr

endmqm

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.

What logs look like

Logs consist of primary and secondary files, and a control file. You define the number and size of log files and where they are stored in the file system.

An IBM MQ log consists of two components:

1. One or more files of log data.
2. A log control file

A file of log data is also known as a log extent.

There are a number of log extents that contain the data being recorded. You can define the number and size (as explained in "Log defaults for IBM MQ" on page 930), 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 1325 for more information.

In IBM MQ for Windows, if you have not changed the log path, log extents are created under the directory:

`C:\ProgramData\IBM\MQ\log\`*`QMgrName`*

In IBM MQ for UNIX and Linux systems, if you have not changed the log path, log extents are created under the directory:

`/var/mqm/log/`*`QMgrName`*

IBM MQ starts with these primary log extents, but if the primary log space is not sufficient, it allocates *secondary* log extents. It does this dynamically and removes them when the demand for log space reduces. By default, up to two secondary log extents can be allocated. You can change this default allocation, as described in "Changing IBM MQ and queue manager configuration information" on page 906.

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 `amqhlctl.lfh`. When backing up or restoring your queue manager, ensure that the log control file is backed up and restored, along with your log extents.

Types of logging

In IBM MQ there are two ways of maintaining records of queue manager activities: circular logging and linear logging.

Circular logging

Use circular logging if all you want is restart recovery, using the log to roll back transactions that were in progress when the system stopped.

Circular logging keeps all restart data in a ring of log files. Logging fills the first file in the ring, then moves on to the next, and so on, until all the files are full. It then goes back to the first file in the ring and starts again. This continues as long as the product is in use, and has the advantage that you never run out of log files.

IBM MQ keeps the log entries required to restart the queue manager without loss of data until they are no longer required to ensure queue manager data recovery. The mechanism for releasing log files for reuse is described in "Using checkpointing to ensure complete recovery" on page 1327.

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.

V 9.0.4 From IBM MQ Version 9.0.4 you can change an existing log type for a queue manager, from linear to circular, or from circular to linear using the migmqlog command.

Multi V 9.0.2
Logger changes

From IBM MQ Version 9.0.2, 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.

ULW 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 1331 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 1337 for further information about the disposition of log files.

Secondary log files

Although secondary log files are defined for linear logging, they are not used in normal operation. If a situation arises when, probably due to long-lived transactions, it is not possible to free a file from the active pool because it might still be required for a restart, secondary files are formatted and added to the active log file pool.

If the number of secondary files available is used up, requests for most further operations requiring log activity will be refused with an MQRC_RESOURCE_PROBLEM return code being returned to the application, and any long running transactions will be considered for asynchronous rollback.

Attention: Both types of logging can cope with unexpected loss of power, assuming that there is no hardware failure.

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 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=00000001E45FF7AD to RBA=00000001E48C1882
CSQR005I +MQOX RESTART - FORWARD RECOVERY COMPLETE -  816
IN COMMIT=0, INDOUBT=0
```

```
CSQR032I +MQOX Backward recovery log range  817
from RBA=00000001E48C1882 to RBA=00000001E48C1882
CSQR006I +MQOX RESTART - BACKWARD RECOVERY COMPLETE -  818
INFLIGHT=0, IN BACKOUT=0
CSQR002I +MQOX RESTART COMPLETED
```

Note: If there is a large amount of log to be read, messages CSQR031I (forward recovery) and CSQR033I (backwards recovery) are issued periodically to show the progression.

In Figure 164, 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.

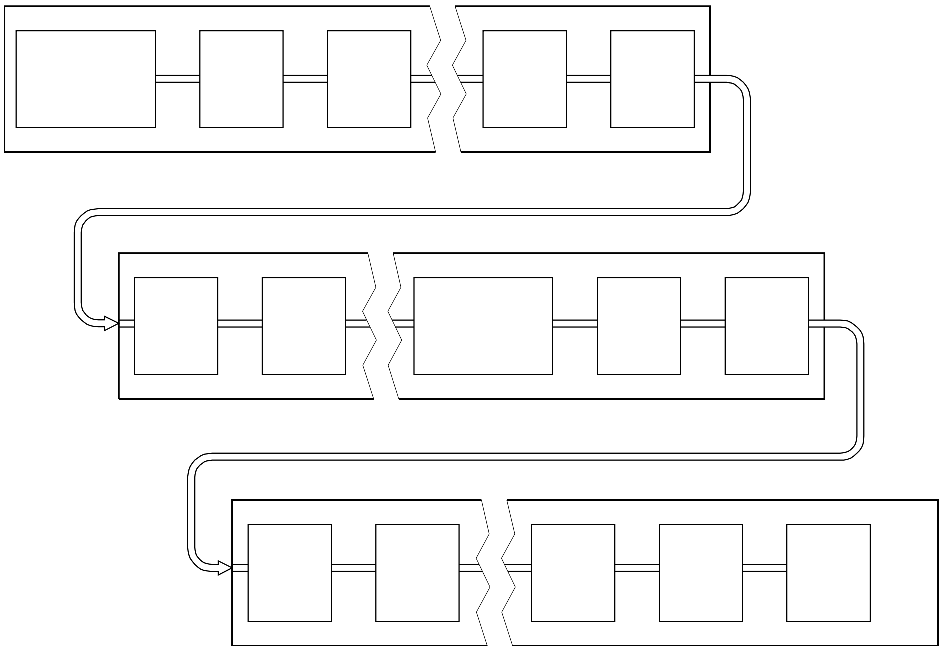


Figure 164. Checkpointing. For simplicity, only the ends of the log files are shown.

Checkpointing with long-running transactions:

How a long-running transaction affects reuse of log files.

Figure 165 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.

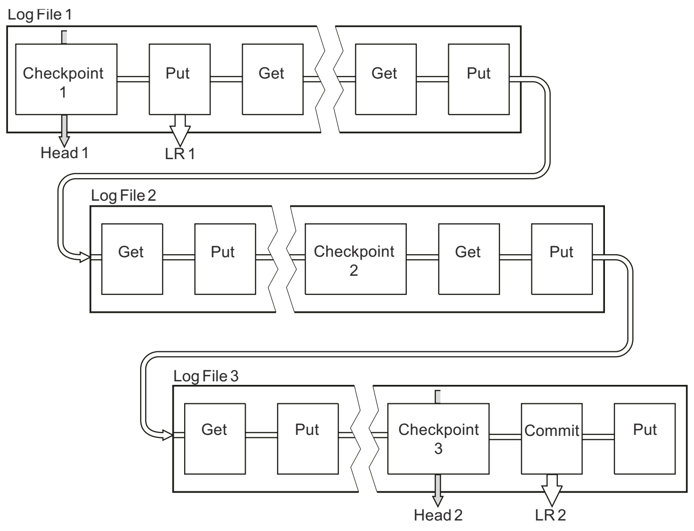


Figure 165. Checkpointing with a long-running transaction. For simplicity, only the ends of the log files are shown.

Calculating the size of the log

Estimating the size of log a queue manager needs.

After deciding whether the queue manager uses circular or linear logging, you need to estimate the size of the Active log that the queue manager needs. The size of the active log is determined by the following log configuration parameters:

LogFilePages
: The size of each primary and secondary log file in units of 4K pages

LogPrimaryFiles
: The number of preallocated primary log files

LogSecondaryFiles
: The number of secondary log files that can be created for use when the primary log files are becoming full

Notes:

1. You can change the number of primary and secondary log files each time the queue manager starts, although you might not notice the effect of the change you make to the secondary logs immediately.
2. You cannot change the log file size; you must determine it **before** creating the queue manager.
3. The number of primary log files and the log file size determine the amount of log space that is preallocated when the queue manager is created.
4. The total number of primary and secondary log files cannot exceed 511 on UNIX and Linux systems, or 255 on Windows, which in the presence of long-running transactions, limits the maximum amount of log space available to the queue manager for restart recovery. The amount of log space the queue manager might need for media recovery does not share this limit.
5. When *circular* logging is being used, the queue manager reuses primary and secondary log space. The queue manager will, up to a limit, allocate a secondary log file when a log file becomes full, and the next primary log file in the sequence is not available.

   See "How large should I make my active log?" on page 1332 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.

   V 9.0.2 From IBM MQ Version 9.0.2 the queue manager attempts to schedule checkpoints in a manner that keeps the log usage within the primary extents.

See "Log defaults for IBM MQ" on page 930 for more information.

How large should I make my active log?:

Estimating the size of active log a queue manager needs.

The size of the active log is limited by:

```
logsize = (primaryfiles + secondaryfiles) * logfilepages * 4096
```

The log should be large enough to cope with your longest running transaction running when the queue manager is writing the maximum amount of data per second to disk.

If your longest running transaction runs for `N` seconds, and the maximum amount of data per second written to disk by the queue manager is `B` bytes per second in the log, your log should be at least:

```
logsize >= 2 * (N+1) * B
```

The queue manager is likely to be writing the maximum amount of data per second to disk when you are running at peak workload, or it might be when you are recording media images.

If a transaction runs for so long that the log extent containing its first log record is not contained within the active log, the queue manager rolls back active transactions one at a time, starting with the transaction with the oldest log record.

The queue manager needs to make old log extents inactive before the maximum number of primary and secondary files are being used, and the queue manager needs to allocate another log extent.

Decide how long you want your longest running transaction to run, before the queue manager is allowed to roll it back. Your longest running transaction might be waiting for slow network traffic or, in the case of a poorly designed transaction, waiting for user input.

You can investigate how long your longest running transaction runs for, by issuing the following **runmqsc** command:

```
DISPLAY CONN(*) UOWLOGDA UOWLOGTI
```

Issuing the `dspmqtrn -a` command, shows all the XA and non XA commands in all states.

Issuing this command lists the date and time that the first log record was written for all of your current transactions.

Attention: For the purposes of calculating the log size, it is the time since the first log record was written that matters, not the time since the application or transaction started. Round up the length of your longest running transaction to the nearest second. This is because of optimizations in the queue manager.

The first log record can be written long after the application started, if the application begins by, for example, issuing an MQGET call that waits for a length of time before actually getting a message.

By reviewing the maximum observed date and time output from the

```
DISPLAY CONN(*) UOWLOGDA UOWLOGTI
```

command you issued originally, from the current date and time, you can estimate how long your longest running transaction runs.

Ensure you run this **runmqsc** command repeatedly while your longest running transactions are running at peak workload so that you do not underestimate the length of your longest running transaction.

In IBM MQ Version 8.0 use the operating system tools, for example, **iostat** on UNIX platforms.

From IBM MQ Version 9.0, 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(414D51436D61726B20202020202020202020)
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.

  V 9.0.2 From IBM MQ Version 9.0.2, 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.

  V 9.0.2 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.
- V 9.0.2 From IBM MQ Version 9.0.2 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 1327 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.

V 9.0.2 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 1341 and ALTER QMGR.

See “Using checkpointing to ensure complete recovery” on page 1327 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
```

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 + 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

V 9.0.2 From Version 9.0.2, IBM MQ 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

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.

V 9.0.2 From IBM MQ Version 9.0.2 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.

Multi If the log fills, message AMQ7463 is issued. In addition, if the log fills because a long-running transaction has prevented the space being released, message AMQ7465 is issued.

Of the log records, only those written since the start of the last complete checkpoint, and those written by any active transactions, are needed to restart the queue manager.

Over time, the oldest log records written become unnecessary for restarting the queue manager.

When a long-running transaction is detected, activity is scheduled to asynchronously rollback that transaction. If, for some unexpected reason, that asynchronous rollback were to fail, some MQI calls return MQRC_RESOURCE_PROBLEM in that situation.

Note that space is reserved to commit or roll back all in-flight transactions, so `MQCMIT` or `MQBACK` should not fail.

The queue manager rolls back transactions that run for a long duration. An application that has a transaction is rolled back in this way cannot perform subsequent `MQPUT` or `MQGET` operations specifying sync point under the same transaction.

However, transactions ended manually start a new log. Note, that whereas new log space is allocated immediately, log space that has been released takes a finite time to be freed up.

An attempt to put or get a message under sync point in this state returns MQRC_BACKED_OUT. The application can then issue **`MQCMIT`**, which returns MQRC_BACKED_OUT, or **`MQBACK`** and start a new transaction. When the transaction consuming too much log space has been rolled back, the log space is released and the queue manager continues to operate normally.

What happens when a disk gets full:

The queue manager logging component can cope with a full disk, and with full log files. If the disk containing the log fills, the queue manager issues message AMQ6709 and an error record is taken.

The log files are created at their fixed size, rather than being extended as log records are written to them. This means that IBM MQ can run out of disk space only when it is creating a new file; it cannot run out of space when it is writing a record to the log. IBM MQ always knows how much space is available in the existing log files, and manages the space within the files accordingly.

V 9.0.2 From IBM MQ Version 9.0.2, 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 “Log defaults for IBM MQ” on page 930 and “Queue manager logs” on page 938). 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 Script (MQSC) Commands, and for information about PCF commands, see Automating administration tasks.

Queue manager messages
Periodically, the queue manager issues a pair of messages to indicate which of the log files are needed:

- Message AMQ7467 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 AMQ7468 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 AMQ6767 is issued. Make the log file, and all subsequent log files, available to the queue manager and try the operation again.

Multi V 9.0.2

Cleaning log extents automatically - linear logging only

From IBM MQ Version 9.0.2 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 "Queue manager logs" on page 938 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

V 9.0.2

Taking media images automatically - linear logging only

From IBM MQ Version 9.0.2 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** will not be moved forward if you are not taking media images, so you need to, either be archiving extents, or be sure you have sufficient disk space.

You can also control automatic and manual media images for other user-defined objects:
- 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. In IBM MQ Version 9.0.2, the queue manager writes persistent messages only in media images, not nonpersistent messages. This can reduce the size of media images when migrating to IBM MQ Version 9.0.2 or later

V 9.0.2

Deciding how to set IMGLOGLN and IMGINTVL

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.

V 9.0.1

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:

Windows **On Windows**

```
rcdmqimg -m QMNAME -t all *
```

UNIX Linux **On UNIX and Linux**

```
rcdmqimg -m QMNAME -t all "*"
```

Running **rcdmqimg** moves the media log sequence number (LSN) forwards. For further details on log sequence numbers, see “Dumping the contents of the log using the dmpmqlog command” on page 1349. **rcdmqimg** does not run automatically, therefore must be run manually or from an automatic task you have created. For more information about this command, see rcdmqimg and dmpmqlog.

Note: Messages AMQ7467 and AMQ7468 can also be issued at the time of running the `rcdmqimg` command.

V 9.0.2

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.

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`.
- V 9.0.2 Determine the name for archive notification, issue the command `DISPLAY QMSTATUS ARCHLOG`.

V 9.0.2 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.

Log file location:

When choosing a location for your log files, remember that operation is severely affected if IBM MQ fails to format a new log because of lack of disk space.

If you are using a circular log, ensure that there is sufficient space on the drive for at least the configured primary log files. Also leave space for at least one secondary log file, which is needed if the log has to grow.

If you are using a linear log, allow considerably more space; the space consumed by the log increases continuously as data is logged.

You should place the log files on a separate disk drive from the queue manager data.

Data integrity on this device is paramount - you should allow for built in redundancy.

It might also be possible to place the log files on multiple disk drives in a mirrored arrangement. This protects against failure of the drive containing the log. Without mirroring, you could be forced to go back to the last backup of your IBM MQ system.

Using the log for recovery

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

V 9.0.2 From IBM MQ Version 9.0.2, on a linear logging queue manager, media images can be recorded only for objects that are recoverable. For example, you need to consider the **`IMGRCOVO`** and **`IMGRCOVQ`** options.

V 9.0.2 Similarly, you can recover a subset of objects only, defined as media recoverable, from their media images on a linear logging queue manager. In the event that an object, that is not defined as media recoverable is damaged, the options for that object are the same as those for a circular logging queue manager.

Media recovery re-creates objects from information recorded in a linear log. For example, if an object file is inadvertently deleted, or becomes unusable for some other reason, media recovery can re-create it. The information in the log required for media recovery of an object is called a *media image*.

A media image is a sequence of log records containing an image of an object from which the object itself can be re-created.

The first log record required to re-create an object is known as its *media recovery record* ; it is the start of the latest media image for the object. The media recovery record of each object is one of the pieces of information recorded during a checkpoint.

When an object is re-created from its media image, it is also necessary to replay any log records describing updates performed on the object since the last image was taken.

Consider, for example, a local queue that has an image of the queue object taken before a persistent message is put onto the queue. In order to re-create the latest image of the object, it is necessary to replay the log entries recording the putting of the message to the queue, in addition to replaying the image itself.

When an object is created, the log records written contain enough information to completely re-create the object. These records make up the first media image of the object. Then, at each shutdown, the queue manager records media images automatically as follows:
- Images of all process objects and queues that are not local
- Images of empty local queues

Media images can also be recorded manually using the **rcdmqimg** command, described in rcdmqimg. This command writes a media image of the IBM MQ object.

V 9.0.2 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

IBM MQ automatically recovers some objects from their media image if it finds that they are corrupted or damaged. In particular, recovery applies to objects found to be damaged during the normal queue manager startup. If any transaction was incomplete when the queue manager last shut down, any queue affected is also recovered automatically in order to complete the startup operation.

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.

V 9.0.1

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 UNIX 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.

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 `qmanager/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. At other times, when object damage is detected, operator message AMQ7472 is issued and most operations using the object fail. If the queue manager object is damaged at any time after the queue manager has started, the queue manager performs a pre-emptive shutdown. When an object has been damaged you can delete it or, if the queue manager is using a linear log, attempt to recover it from its media image using the `rcrmqobj` command (see rcrmqobj for further details).

V 9.0.2 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.

V 9.0.2 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.

V 9.0.2 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.

V 9.0.2 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**.

V 9.0.2 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.

Protecting IBM MQ log files

Do not touch the log files when a queue manager is running, recovery might be impossible. Use super user or mqm authority to protect log files against inadvertent modification.

Do not remove the active log files manually when an IBM MQ queue manager is running. If a user inadvertently deletes the log files that a queue manager needs to restart, IBM MQ **does not** issue any errors and continues to process data *including persistent messages*. The queue manager shuts down normally, but can fail to restart. Recovery of messages then becomes impossible.

Users with the authority to remove logs that are being used by an active queue manager also have authority to delete other important queue manager resources (such as queue files, the object catalog, and IBM MQ executable files). They can therefore damage, perhaps through inexperience, a running or dormant queue manager in a way against which IBM MQ cannot protect itself.

Exercise caution when conferring super user or mqm authority.

Dumping the contents of the log using the dmpmqlog command

How to use the `dmpmqlog` command to dump the contents of the queue manager log.

Use the `dmpmqlog` command to dump the contents of the queue manager log. By default all active log records are dumped, that is, the command starts dumping from the head of the log (usually the start of the last completed checkpoint).

The log can usually be dumped only when the queue manager is not running. Because the queue manager takes a checkpoint during shutdown, the active portion of the log usually contains a small number of log records. However, you can use the `dmpmqlog` command to dump more log records using one of the following options to change the start position of the dump:

- Start dumping from the *base* of the log. The base of the log is the first log record in the log file that contains the head of the log. The amount of additional data dumped in this case depends on where the head of the log is positioned in the log file. If it is near the start of the log file, only a small amount of additional data is dumped. If the head is near the end of the log file, significantly more data is dumped.
- Specify the start position of the dump as an individual log record. Each log record is identified by a unique *log sequence number (LSN)*. In the case of circular logging, this starting log record cannot be before the base of the log; this restriction does not apply to linear logs. You might need to reinstate inactive log files before running the command. You must specify a valid LSN, taken from previous `dmpmqlog` output, as the start position.

  For example, with linear logging you can specify the `nextlsn` from your last `dmpmqlog` output. The `nextlsn` appears in `Log File Header` and indicates the LSN of the next log record to be written. Use this as a start position to format all log records written since the last time the log was dumped.
- **For linear logs only**, you can instruct `dmpmqlog` to start formatting log records from any given log file extent. In this case, `dmpmqlog` expects to find this log file, and each successive one, in the same directory as the active log files. This option does not apply to circular logs, where `dmpmqlog` cannot access log records prior to the base of the log.

The output from the `dmpmqlog` command is the `Log File Header` and a series of formatted log records. The queue manager uses several log records to record changes to its data.

Some of the information that is formatted is only of use internally. The following list includes the most useful log records:

`Log File Header`

Each log has a single log file header, which is always the first thing formatted by the `dmpmqlog` command. It contains the following fields:

logactive	The number of primary log extents.
loginactive	The number of secondary log extents.
logsize	The number of 4 KB pages per extent.
baselsn	The first LSN in the log extent containing the head of the log.
nextlsn	The LSN of the next log record to be written.
headlsn	The LSN of the log record at the head of the log.
tailsn	The LSN identifying the tail position of the log.
hflag1	Whether the log is CIRCULAR or LOG RETAIN (linear).
HeadExtentID	The log extent containing the head of the log.

`Log Record Header`

Each log record within the log has a fixed header containing the following information:

LSN	The log sequence number.
LogRecdType	The type of the log record.
XTranid	The transaction identifier associated with this log record (if any). A *TranType* of MQI indicates an IBM MQ-only transaction. A *TranType* of XA is involved with other resource managers. Updates involved within the same unit of work have the same *XTranid*.
QueueName	The queue associated with this log record (if any).
Qid	The unique internal identifier for the queue.
PrevLSN	The LSN of the previous log record within the same transaction (if any).

Start Queue Manager

This logs that the queue manager has started.

StartDate	The date that the queue manager started.
StartTime	The time that the queue manager started.

Stop Queue Manager

This logs that the queue manager has stopped.

StopDate	The date that the queue manager stopped.
StopTime	The time that the queue manager stopped.
ForceFlag	The type of shutdown used.

Start Checkpoint

This denotes the start of a queue manager checkpoint.

End Checkpoint

This denotes the end of a queue manager checkpoint.

ChkPtLSN	The LSN of the log record that started this checkpoint.

Put Message

This logs a persistent message put to a queue. If the message was put under sync point, the log record header contains a non-null *XTranid*. The remainder of the record contains:

MapIndex	An identifier for the message on the queue. It can be used to match the corresponding `MQGET` that was used to get this message from the queue. In this case a subsequent *Get Message* log record can be found containing the same `QueueName` and `MapIndex`. At this point the `MapIndex` identifier can be reused for a subsequent put message to that queue.
Data	Contained in the hex dump for this log record is various internal data, followed by a representation of the Message Descriptor (eyecatcher MD) and then the message data itself.

Put Part

Persistent messages that are too large for a single log record are logged as multiple *Put Part* log records followed by a single *Put Message* record. If there are *Put Part* records, then the *PrevLSN* field will chain the *Put Part* records and the final *Put Message* record together.

Data	Continues the message data where the previous log record left off.

Get Message
Only gets of persistent messages are logged. If the message was got under sync point, the log record header contains a non-null `XTranid`. The remainder of the record contains:

MapIndex	Identifies the message that was retrieved from the queue. The most recent `Put Message` log record containing the same `QueueName` and `MapIndex` identifies the message that was retrieved.
QPriority	The priority of the message retrieved from the queue.

Start Transaction
Indicates the start of a new transaction. A TranType of MQI indicates an IBM MQ-only transaction. A TranType of XA indicates one that involves other resource managers. All updates made by this transaction will have the same `XTranid`.

Prepare Transaction
Indicates that the queue manager is prepared to commit the updates associated with the specified `XTranid`. This log record is written as part of a two-phase commit involving other resource managers.

Commit Transaction
Indicates that the queue manager has committed all updates made by a transaction.

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:

XTranid	The transaction identifier.
FirstLSN	The LSN of the first log record associated with the transaction.
LastLSN	The LSN of the last log record associated with the transaction.

Transaction Participants
This log record is written by the XA Transaction Manager component of the queue manager. It records the external resource managers that are participating in transactions. For each participant the following is recorded:

RMName	The name of the resource manager.
RMID	The resource manager identifier. This is also logged in subsequent `Transaction Prepared` log records that record global transactions in which the resource manager is participating.
SwitchFile	The switch load file for this resource manager.
XAOpenString	The XA open string for this resource manager.
XACloseString	The XA close string for this resource manager.

Transaction Prepared
This log record is written by the XA Transaction Manager component of the queue manager. It indicates that the specified global transaction has been successfully prepared. Each of the participating resource managers will be instructed to commit. The `RMID` of each prepared resource manager is recorded in the log record. If the queue manager itself is participating in the transaction a `Participant Entry` with an `RMID` of zero will be present.

`Transaction Forget`
This log record is written by the XA Transaction Manager component of the queue manager. It follows the *`Transaction Prepared`* log record when the commit decision has been delivered to each participant.

`Purge Queue`
This logs the fact that all messages on a queue have been purged, for example, using the MQSC command CLEAR QUEUE.

`Queue Attributes`
This logs the initialization or change of the attributes of a queue.

`Create Object`
This logs the creation of an IBM MQ object.

ObjName	The name of the object that was created.
UserId	The user ID performing the creation.

`Delete Object`
This logs the deletion of an IBM MQ object.

ObjName	The name of the object that was deleted.

Backing up and restoring IBM MQ queue manager data

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

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."
  - "Restoring queue manager data" on page 1354.
- To back up and restore the queue manager configuration see:
  - Multi "Backing up queue manager configuration" on page 1355
  - Multi "Restoring queue manager configuration" on page 1355
- To create, update and start a backup queue manager, see "Using a backup queue manager" on page 1356.

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 and queue manager configuration information" on page 906.

   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 1324, and the configuration files as described in "Initialization and configuration files" on page 1011. 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.

   UNIX Linux For IBM MQ for UNIX and Linux systems, you can do this with the **`tar`** command. (If you have queues larger than 2 GB, you cannot use the **`tar`** command. For more information, see Enabling large queues.

   Note: When you upgrade to IBM WebSphere MQ Version 7.5 and later, ensure to 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 of IBM MQ.

Related tasks:

"Stopping a queue manager" on page 841
You can use the **endmqm** command to stop a queue manager. This command provides three ways to stop a queue manager: a controlled, or quiesced, shutdown, an immediate shutdown, and a preemptive shutdown. Alternatively, on Windows and Linux, you can stop a queue manager by using the IBM MQ Explorer.

"Backing up configuration files after creating a queue manager" on page 839
IBM MQ configuration information is stored in configuration files on UNIX, 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.

Note: When you upgrade to IBM WebSphere MQ Version 7.5 and later, 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 UNIX and Linux systems.

Results

If the data was backed up and restored correctly, the queue manager will now start.

Backing up queue manager configuration

Multi

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

ULW On UNIX, Linux, and Windows, you can use the **dmpmqcfg** command to dump the configuration of an IBM MQ queue manager.

IBM i 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:
   - ULW On UNIX, Linux, and Windows: Execute the Dump MQ Configuration command, **dmpmqcfg**, using the default formatting option of (-f mqsc) MQSC and all attributes (-a), use standard output redirection to store the definitions into a file. For example:
     ```
     dmpmqcfg -m MYQMGR -a > /mq/backups/MYQMGR.mqsc
     ```
   - IBM i On IBM i: Execute the Dump MQ Configuration command (**DMPMQMCFG**) using the default formatting option of OUTPUT(*MQSC) and EXPATTR(*ALL), use the TOFILE and TOMBR to store the definitions into a physical file member. For example:
     ```
     DMPMQMCFG MQMNAME(MYQMGR) OUTPUT(*MQSC) EXPATTR(*ALL) TOFILE(QMQMSAMP/QMQSC) TOMBR(MYQMGRDEF)
     ```

Related tasks:

"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.

Related information:

dmpmqcfg (dump queue manager configuration)

Dump MQ Configuration (DMPMQMCFG)

Restoring queue manager configuration

Multi

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

ULW On UNIX, 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:
   - ULW On UNIX, 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 (**dmpmqcfg**) command (see "Backing up queue manager configuration" on page 1355). 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 1355). For example:
     ```
     STRMQMMQSC MQMNAME(MYQMGR) SRCFILE(QMQMSAMP/QMQSC) SRCMBR(MYQMGR)
     ```

Related tasks:

"Backing up queue manager configuration" on page 1355
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 information:

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. 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 how to create, update, and start a backup queue manager, see the following topics:
- "Creating a backup queue manager"
- "Updating a backup queue manager" on page 1358
- "Starting a backup queue manager" on page 1358

Related concepts:
"Logging: Making sure that messages are not lost" on page 1323
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 1353.
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 information:

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.

Note: 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. 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 information:

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

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 1003
When a channel goes into STOPPED state, you have to restart the channel manually.

Related information:

strmqm (start queue manager)

Configuring JMS resources

One of the ways in which a JMS application can create and configure the resources that it needs to connect to IBM MQ and access destinations for sending or receiving messages is by using the Java Naming and Directory Interface (JNDI) to retrieve administered objects from a location within the naming and directory service that is called the JNDI namespace. Before a JMS application can retrieve administered objects from a JNDI namespace, you must first create and configure the administered objects.

About this task

You can create and configure administered objects in IBM MQ by using either of the following tools:

IBM MQ Explorer
: You can use IBM MQ Explorer to create and administer JMS object definitions that are stored in LDAP, in a local file system, or other locations.

IBM MQ JMS administration tool
: The IBM MQ JMS administration tool is a command-line tool that you can use to create and

configure IBM MQ JMS objects that are stored in LDAP, in a local file system, or other locations. The JMS administration tool uses a syntax that is similar to **runmqsc**, and also supports scripting.

The administration tool uses a configuration file to set the values of certain properties. A sample configuration file is supplied, which you can edit to suit your system before you start by using the tool to configure JMS resources. For more information about the configuration file, see "Configuring the JMS administration tool" on page 1366.

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.

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 information:

Writing IBM MQ classes for JMS applications

runmqsc

Configuring connection factories and destinations in a JNDI namespace

JMS applications access administered objects in the naming and directory service through the Java Naming and Directory Interface (JNDI). The JMS administered objects are stored in a location within the naming and directory service that is referred to as the JNDI namespace. A JMS application can look up the administered objects to connect to IBM MQ and access destinations for sending or receiving messages.

About this task

JMS applications look up the names of the JMS objects in the naming and directory service by using contexts:

Initial context
The initial context defines the root of the JNDI namespace. For each location in the naming and directory service, you need to specify an initial context to give a starting point from which a JMS application can resolve the names of the administered objects in that location of the naming and directory service.

Subcontexts
A context can have one or more subcontexts. A subcontext is a subdivision of a JNDI namespace and can contain administered objects such as connection factories and destinations as well as other subcontexts. A subcontext is not an object in its own right; it is merely an extension of the naming convention for the objects in the subcontext.

You can create contexts using either IBM MQ Explorer or the IBM MQ JMS administration tool.

Before an IBM MQ classes for JMS application can retrieve administered objects from a JNDI namespace, you must first create the administered objects using either IBM MQ Explorer or the IBM MQ JMS administration tool. You can create and configure the following types of JMS object:

Connection factory
A JMS connection factory object defines a set of standard configuration properties for connections. A JMS application uses a connection factory to create a connection to IBM MQ. You can create a connection factory that is specific to one of the two messaging domains, the point-to-point messaging domain and the publish/subscribe messaging domain. Alternatively, from JMS 1.1, you can create domain-independent connection factories that can be used for both point-to-point and publish/subscribe messaging.

Destination
A JMS destination is an object that represents the target of messages that the client produces and the source of messages that a JMS application consumes. The JMS application can either use a single destination object to put messages on and to get messages from, or the application can use separate destination objects. There are two types of destination object:
- JMS queue destination used in point-to-point messaging
- JMS topic destination used in publish/subscribe messaging

The following diagram shows an example of JMS objects created in an IBM MQ JNDI namespace.

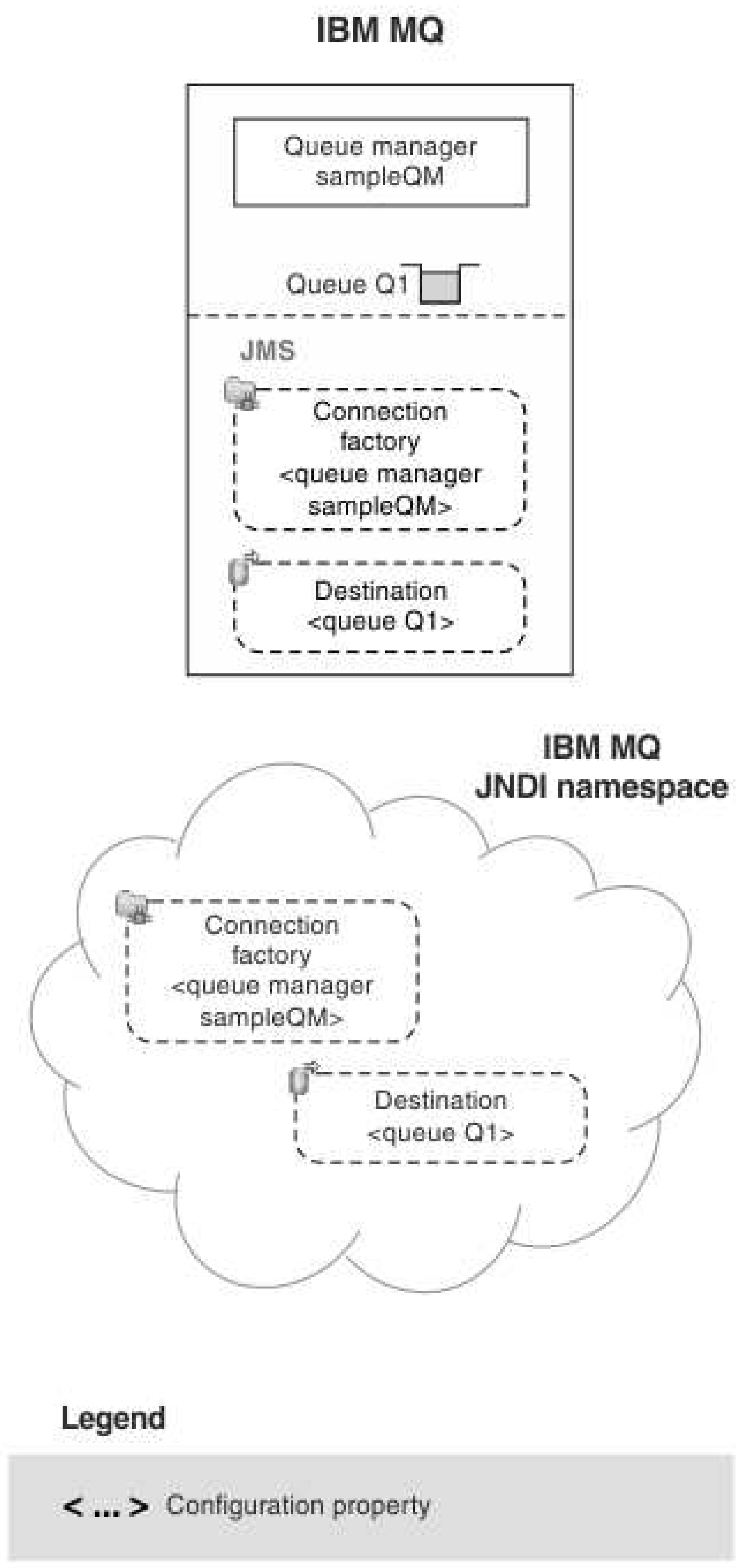

IBM MQ
Queue manager sampleQM
Queue Q1
JMS
Connection factory <queue manager sampleQM>
Destination <queue Q1>
IBM MQ JNDI namespace
Connection factory <queue manager sampleQM>
Destination <queue Q1>
Legend
< ... > Configuration property

Figure 166. JMS objects created in IBM MQ

If you use JMS messaging between WebSphere Application Server and IBM MQ, you must create corresponding objects in WebSphere Application Server to use to communicate with IBM MQ. When you create one of these objects in WebSphere Application Server, it is stored in the WebSphere Application Server JNDI namespace as shown in the following diagram.

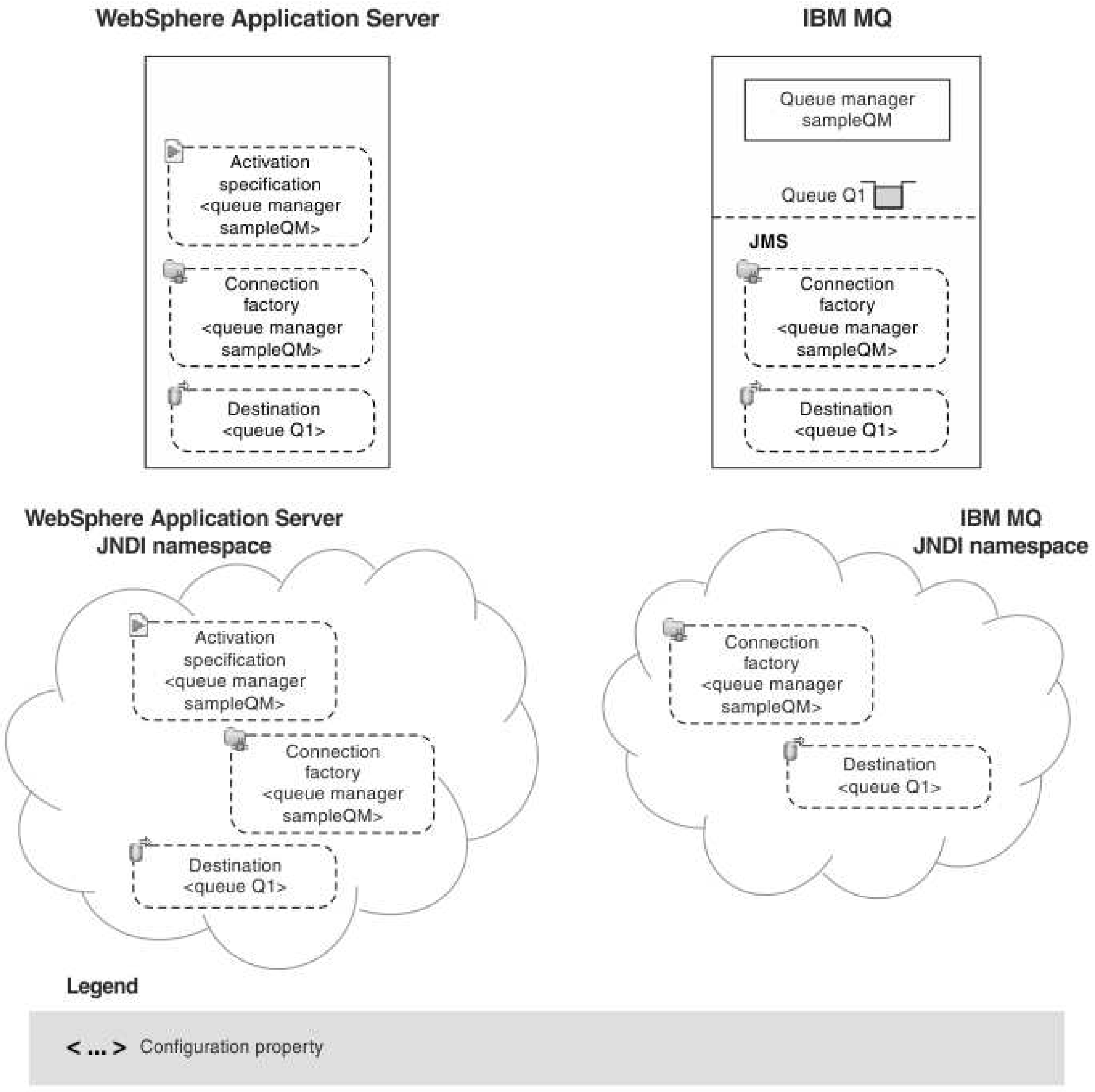

WebSphere Application Server
Activation specification <queue manager sampleQM>
Connection factory <queue manager sampleQM>
Destination <queue Q1>
IBM MQ
Queue manager sampleQM
Queue Q1
JMS
Connection factory <queue manager sampleQM>
Destination <queue Q1>
WebSphere Application Server JNDI namespace
Activation specification <queue manager sampleQM>
Connection factory <queue manager sampleQM>
Destination <queue Q1>
IBM MQ JNDI namespace
Connection factory <queue manager sampleQM>
Destination <queue Q1>
Legend
<...> Configuration property

Figure 167. Objects created in WebSphere Application Server, and the corresponding objects in IBM MQ

If your application uses a message-driven bean (MDB), the connection factory is used for outbound messages only and inbound messages are received by an activation specification. Activation specifications are part of the Java EE Connector Architecture 1.5 (JCA 1.5) standard. JCA 1.5 provides a standard way to integrate JMS providers, such as IBM MQ, with Java EE application servers such as WebSphere Application Server. A JMS activation specification can be associated with one or more message driven beans (MDBs) and provides the configuration necessary for these MDBs to listen for messages arriving at a destination.

You can use either the WebSphere Application Server administrative console or wsadmin scripting commands to create and configure the JMS resources that you need.

Procedure

- To configure JMS objects for IBM MQ using IBM MQ Explorer, see "Configuring JMS objects using IBM MQ Explorer."
- To configure JMS objects for IBM MQ using the IBM MQ JMS administration tool, see "Configuring JMS objects using the administration tool" on page 1365.
- To configure JMS objects for WebSphere Application Server, see "Configuring JMS resources in WebSphere Application Server" on page 1373.

Results

An IBM MQ classes for JMS application can retrieve the administered objects from the JNDI namespace and, if required, set or change one or more of its properties by using either the IBM JMS extensions or the IBM MQ JMS extensions.

Related information:

Using JNDI to retrieve administered objects in a JMS application

Creating and configuring connection factories and destinations in an IBM MQ classes for JMS application

Configuring JMS objects using 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

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 x86-64. It can remotely connect to queue managers that are running on any supported platform including z/OS, enabling your entire messaging backbone to be viewed, explored, and altered from the console.

In 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 information:

Introduction to IBM MQ Explorer

Configuring JMS objects using the administration tool

You can use the IBM MQ JMS administration tool to define the properties of eight types of IBM MQ classes for JMS object and to store them within a JNDI namespace. Applications can then use JNDI to retrieve these administered objects from the namespace.

About this task

The following table shows the eight types of administered objects that you can create, configure and manipulate using verbs. The Keyword column shows the strings that you can substitute for `TYPE` in the commands shown in Table 131.

Table 131. The JMS object types that are handled by the administration tool

Object Type	Keyword	Description
MQConnectionFactory	CF	The IBM MQ implementation of the JMS ConnectionFactory interface. This represents a factory object for creating connections in both the point-to-point and publish/subscribe domains.
MQQueueConnectionFactory	QCF	The IBM MQ implementation of the JMS QueueConnectionFactory interface. This represents a factory object for creating connections in the point-to-point domain.
MQTopicConnectionFactory	TCF	The IBM MQ implementation of the JMS TopicConnectionFactory interface. This represents a factory object for creating connections in the publish/subscribe domain.
MQQueue	Q	The IBM MQ implementation of the JMS Queue interface. This represents a destination for messages in the point-to-point domain.
MQTopic	T	The IBM MQ implementation of the JMS Topic interface. This represents a destination for messages in the publish/subscribe domain.
MQXAConnectionFactory [1]	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]	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.

Table 131. The JMS object types that are handled by the administration tool (continued)

Object Type	Keyword	Description
MQXATopicConnectionFactory [1]	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 1373.

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 1370.

You can also create and configure JMS administered objects with IBM MQ Explorer.

Related information:

Creating and configuring connection factories and destinations in an IBM MQ classes for JMS application

Using JNDI to retrieve administered objects in a JMS application

Configuring the JMS administration tool

The IBM MQ JMS administration tool uses a configuration file to set the values of certain properties. A sample configuration file is supplied, which you can edit to suit your system.

About this task

The configuration file is a plain-text file that consists of a set of key-value pairs, separated by the equal sign (=). You configure the administration tool by setting values for the three properties defined in the configuration file. The following example shows these three properties:

```
#Set the service provider
INITIAL_CONTEXT_FACTORY=com.sun.jndi.ldap.LdapCtxFactory
#Set the initial context
PROVIDER_URL=ldap://polaris/o=ibm_us,c=us
#Set the authentication type
SECURITY_AUTHENTICATION=none
```

(In this example, a hash sign (#) in the first column of the line indicates a comment, or a line that is not used.)

A sample configuration file, which is used as the default configuration file, is supplied with IBM MQ. The sample file is called `JMSAdmin.config`, and is found in the *`MQ_JAVA_INSTALL_PATH`*`/bin` directory. You can either edit this sample file to define the settings needed for your system, or create your own configuration file.

When you start the administration tool, you can specify the configuration file that you want to use by using the `-cfg` command-line parameter, as described in "Starting the administration tool" on page 1368. If you do not specify a configuration file name when you invoke the tool, the tool attempts to load the default configuration file (`JMSAdmin.config`). It searches for this file first in the current directory, and then in the *`MQ_JAVA_INSTALL_PATH`*`/bin` directory, where *`MQ_JAVA_INSTALL_PATH`* is the path to your IBM MQ classes for JMS installation.

The names of JMS objects that are stored in an LDAP environment must comply with LDAP naming conventions. One of these conventions is that object and context names must include a prefix, such as `cn=` (common name), or `ou=` (organizational unit). The administration tool simplifies the use of LDAP service providers by allowing you to refer to object and context names without a prefix. If you do not supply a prefix, the tool automatically adds a default prefix to the name you supply. For LDAP, this is `cn=`. If required, you can change the default prefix by setting the **`NAME_PREFIX`** property in the configuration file.

Note: You might need to configure your LDAP server to store Java objects. For more information, see the documentation for your LDAP server.

Procedure

1. Define the service provider that the tool uses by configuring the **`INITIAL_CONTEXT_FACTORY`** property. The supported values for this property are as follows:
   - com.sun.jndi.ldap.LdapCtxFactory (for LDAP)
   - com.sun.jndi.fscontext.RefFSContextFactory (for file system context)
   - z/OS com.ibm.jndi.LDAPCtxFactory is supported on z/OS only, and provides access to an LDAP server. However, this class is incompatible with com.sun.jndi.ldap.LdapCtxFactory, in that objects created using one InitialContextFactory cannot be read or modified using the other.

   You can also use the administration tool to connect to other JNDI contexts by using three parameters defined in the JMSAdmin configuration file. To use a different InitialContextFactory:
   a. Set the **`INITIAL_CONTEXT_FACTORY`** property to the required class name.
   b. Define the behavior of the InitialContextFactory using the **`USE_INITIAL_DIR_CONTEXT`**, **`NAME_PREFIX`** and **`NAME_READABILITY_MARKER`** properties. The settings for these properties are described in the sample configuration file comments.

   You do not need to define the **`USE_INITIAL_DIR_CONTEXT`**, **`NAME_PREFIX`** and **`NAME_READABILITY_MARKER`** properties if you use one of the supported **`INITIAL_CONTEXT_FACTORY`** values. However, you can give values to these properties if you want to override the system defaults. For example, if your objects are stored in an LDAP environment, you can change the default prefix that the tool adds to object and context names by setting the **`NAME_PREFIX`** property to the required prefix.

   If you omit one or more of the three InitialContextFactory properties, the administration tool provides suitable defaults based on the values of the other properties.
2. Define the URL of the initial context of the session by configuring the **`PROVIDER_URL`** property. This URL is the root of all JNDI operations carried out by the tool. Two forms of this property are supported:
   - ldap://hostname/contextname
   - file:[drive:]/pathname

   The format of the LDAP URL can vary, depending on your LDAP provider. See your LDAP documentation for more information.
3. Define whether JNDI passes security credentials to your service provider by configuring the **`SECURITY_AUTHENTICATION`** property. This property is used only when an LDAP service provider is used and can take one of three values:

   `none` (anonymous authentication)
   If you set this parameter to `none`, JNDI does not pass any security credentials to the service provider, and *anonymous authentication* is performed.

`simple` (simple authentication)
: If you set the parameter to `simple`, security credentials are passed through JNDI to the underlying service provider. These security credentials are in the form of a user distinguished name (User DN) and password.

`CRAM-MD5` (CRAM-MD5 authentication mechanism)
: If you set the parameter to `CRAM-MD5`, security credentials are passed through JNDI to the underlying service provider. These security credentials are in the form of a user distinguished name (User DN) and password.

If you do not supply a valid value for the **`SECURITY_AUTHENTICATION`** property, the property defaults to `none`.

If security credentials are required, you are prompted for them when the tool initializes. You can avoid this by setting the **`PROVIDER_USERDN`** and **`PROVIDER_PASSWORD`** properties in the JMSAdmin configuration file.

Note: If you do not use these properties, the text typed, *including the password*, is echoed to the screen. This might have security implications.

The tool does no authentication itself; the authentication task is delegated to the LDAP server. The LDAP server administrator must set up and maintain access privileges to different parts of the directory. See your LDAP documentation for more information. If authentication fails, the tool displays an appropriate error message and terminates.

More detailed information about security and JNDI is in the documentation at Oracle's Java website (Oracle Technology Network for Java Developers).

Starting the administration tool

The administration tool has a command-line interface that you can use either interactively, or to start a batch process.

About this task

The interactive mode provides a command prompt where you can enter administration commands. In the batch mode, the command to start the tool includes the name of a file that contains an administration command script.

Procedure

Interactive mode

- To start the tool in interactive mode, enter the following command:

  ```
  JMSAdmin [-t] [-v] [-cfg config_filename]
  ```

  where:

  -t
  : Enables trace (default is trace off)

    The trace file is generated in `"%MQ_JAVA_DATA_PATH%"\errors` (Windows) or `/var/mqm/trace` (UNIX). The name of the trace file is of the form:

    ```
    mqjms_PID.trc
    ```

    where *`PID`* is the process ID of the JVM.

  -v
  : Produces verbose output (default is terse output)

  -cfg config_filename
  : Names an alternative configuration file. If this parameter is omitted, the default configuration file, `JMSAdmin.config`, is used. For more information about the configuration file, see "Configuring the JMS administration tool" on page 1366.

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 JMS administration tool” on page 1366.

As you traverse the directory namespace, the prompt changes to reflect this, so that the prompt always displays the current context.

Batch mode

- To start the tool in batch mode, enter the following command:

  ```
  JMSAdmin test.scp
  ```

  where *test.scp* is a script file that contains administration commands. For more information, see “Using administration commands.” The last command in the file must be the END command.

Using administration commands

The administration tool accepts commands consisting of an administration verb and its appropriate parameters.

About this task

The following table lists the administration verbs that you can use when entering commands with the administration tool.

Table 132. 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 administration tool” on page 1368. 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 1370.

- Enter commands in the following form:

  ```
  verb [param]*
  ```

where **verb** is one of the administration verbs listed in Table 132 on page 1369. All valid commands contain one verb, which appears at the beginning of the command in either its standard or short form. Verb names are not case-sensitive.

- To terminate a command, press Enter, unless you want to enter several commands together, in which case type the plus sign (+) directly before pressing Enter. Typically, to terminate commands, you press Enter. However, you can override this by typing the plus sign (+) directly before pressing Enter. This enables you to enter multiline commands, as shown in the following example:

```
DEFINE Q(BookingsInputQueue) +
QMGR(QM.POLARIS.TEST) +
QUEUE(BOOKINGS.INPUT.QUEUE) +
PORT(1415) +
CCSID(437)
```

- To close the administration tool, use the **END** verb. This verb cannot take any parameters.

Configuring subcontexts

You can use the verbs **CHANGE**, **DEFINE**, **DISPLAY** and **DELETE** to configure directory namespace subcontexts.

About this task

The use of these verbs is described in the following table.

Table 133. Syntax and description of commands used to manipulate subcontexts

Command syntax	Description
DEFINE CTX(ctxName)	Attempts to create a child subcontext of the current context, having the name ctxName. Fails if there is a security violation, if the subcontext already exists, or if the name supplied is not valid.
DISPLAY CTX	Displays the contents of the current context. Administered objects are annotated with `a`, subcontexts with `[D]`. The Java type of each object is also displayed.
DELETE CTX(ctxName)	Attempts to delete the current context's child context having the name ctxName. Fails if the context is not found, is non-empty, or if there is a security violation.
CHANGE CTX(ctxName)	Alters the current context, so that it now refers to the child context having the name ctxName. One of two special values of ctxName can be supplied: **=UP** moves to the parent of the current context **=INIT** moves directly to the initial context Fails if the specified context does not exist, or if there is a security violation.

The names of JMS objects that are stored in an LDAP environment must comply with LDAP naming conventions. One of these conventions is that object and context names must include a prefix, such as `cn=` (common name), or `ou=` (organizational unit). The administration tool simplifies the use of LDAP service providers by allowing you to refer to object and context names without a prefix. If you do not supply a prefix, the tool automatically adds a default prefix to the name you supply. For LDAP, this is `cn=`. If required, you can change the default prefix by setting the **NAME_PREFIX** property in the configuration file. For more information, see “Configuring the JMS administration tool” on page 1366.

Note: You might need to configure your LDAP server to store Java objects. For more information, see the documentation for your LDAP server.

Creating JMS objects

To create JMS connection factory and destination objects and store them in a JNDI namespace, use the `DEFINE` verb. To store your objects in an LDAP environment, you must give them names that comply with certain conventions. The administration tool can help you obey LDAP naming conventions by adding a default prefix to object names.

About this task

The DEFINE verb creates an administered object with the type, name and properties that you specify. The new object is stored in the current context.

The names of JMS objects that are stored in an LDAP environment must comply with LDAP naming conventions. One of these conventions is that object and context names must include a prefix, such as `cn=` (common name), or `ou=` (organizational unit). The administration tool simplifies the use of LDAP service providers by allowing you to refer to object and context names without a prefix. If you do not supply a prefix, the tool automatically adds a default prefix to the name you supply. For LDAP, this is `cn=`. If required, you can change the default prefix by setting the **`NAME_PREFIX`** property in the configuration file. For more information, see "Configuring the JMS administration tool" on page 1366.

Note: You might need to configure your LDAP server to store Java objects. For more information, see the documentation for your LDAP server.

Procedure

1. If the administration tool is not already started, start it as described in "Starting the administration tool" on page 1368. The command prompt is displayed, indicating that the tool is ready to accept administration commands.
2. Make sure that command prompt is showing the context in which you want to create the new object. When you start the administration tool, the prompt initially appears as:
   ```
   InitCtx>
   ```
   To change the current context, use the `CHANGE` verb as described in "Configuring subcontexts" on page 1370.
3. To create a connection factory, queue destination or topic destination, use the following command syntax:
   ```
   DEFINE TYPE (name) [property]*
   ```
   That is, type the `DEFINE` verb, followed by a *`TYPE`* `(name)` administered object reference, followed by zero or more *properties* (see Properties of IBM MQ classes for JMS objects).
4. To create a connection factory, queue destination or topic destination, use the following command syntax:
   ```
   DEFINE TYPE (name) [property]*
   ```
5. To display the newly created object, use the `DISPLAY` verb with the following command syntax:
   ```
   DISPLAY TYPE (name)
   ```

Example

The following example shows a queue called testQueue created in the initial context using the `DEFINE` verb. Since this object is being stored in an LDAP environment, although the object name `testQueue` is not entered with a prefix, the tool automatically adds one to ensure compliance with the LDAP naming convention. Submitting the command `DISPLAY Q(testQueue)` also causes this prefix to be added.

```
InitCtx> DEFINE Q(testQueue)

InitCtx> DISPLAY CTX

Contents of InitCtx
```

```
a cn=testQueue       com.ibm.mq.jms.MQQueue

1 Object(s)
0 Context(s)
1 Binding(s), 1 Administered
```

Sample error conditions creating a JMS object:

A number of common error conditions can arise when you create an object.

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 134 summarizes the use of these verbs. Substitute *`TYPE`* with the keyword that represents the required administered object, as described in "Configuring JMS objects using the administration tool" on page 1365.

Table 134. Syntax and description of commands used to manipulate administered objects

Command syntax	Description
ALTER *TYPE* (name) [property]*	Attempts to update the properties of the administered object with the ones supplied. Fails if there is a security violation, if the specified object cannot be found, or if the new properties supplied are not valid.
DEFINE *TYPE* (name) [property]*	Attempts to create an administered object of type *`TYPE`* with the supplied properties, and store it under the name `name` in the current context. Fails if there is a security violation, if the supplied name is not valid or an object of that name exists, or if the properties supplied are not valid.
DISPLAY *TYPE* (name)	Displays the properties of the administered object of type *`TYPE`* , bound under the name `name` in the current context. Fails if the object does not exist, or if there is a security violation.
DELETE *TYPE* (name)	Attempts to remove the administered object of type *`TYPE`* , having the name `name`, from the current context. Fails if the object does not exist, or if there is a security violation.
COPY *TYPE* (nameA) *TYPE* (nameB)	Makes a copy of the administered object of type *`TYPE`* , having the name `nameA`, naming the copy `nameB`. This all occurs within the scope of the current context. Fails if the object to be copied does not exist, if an object of name nameB exists, or if there is a security violation.
MOVE *TYPE* (nameA) *TYPE* (nameB)	Moves (renames) the administered object of type *`TYPE`* , having the name `nameA`, to `nameB`. This all occurs within the scope of the current context. Fails if the object to be moved does not exist, if an object of name nameB exists, or if there is a security violation.

Configuring JMS resources in WebSphere Application Server

To configure JMS resources in WebSphere Application Server, you can either use the administrative console or wsadmin commands.

About this task

Java Message Service (JMS) applications typically rely on externally configured objects which describe how the application connects to its JMS provider and the destinations it accesses. JMS applications use the Java Naming Directory Interface (JNDI) to access the following types of object at runtime:
- Activation specifications (used by Java EE application servers)
- Unified connection factories (with JMS 1.1, domain-independent (unified) connection factories are preferred to domain-specific queue connection factories and topic connection factories)
- Topic connection factories (used by JMS 1.0 applications)
- Queue connection factories (used by JMS 1.0 applications)
- Queues
- Topics

Through the IBM MQ messaging provider in WebSphere Application Server, Java Message Service (JMS) messaging applications can use your IBM MQ system as an external provider of JMS messaging resources. To enable this approach, you configure the IBM MQ messaging provider in WebSphere Application Server to define JMS resources for connecting to any queue manager on the IBM MQ network.

You can use WebSphere Application Server to configure IBM MQ resources for applications (for example queue connection factories) and to manage messages and subscriptions associated with JMS destinations. You administer security through IBM MQ.

Related information for WebSphere Application Server Version 8.5.5

Interoperation using the IBM MQ messaging provider

Managing messaging with the IBM MQ messaging provider

Mapping of administrative console panel names to command names and IBM MQ names

Related information for WebSphere Application Server Version 8.0

Interoperation using the IBM MQ messaging provider

Managing messaging with the IBM MQ messaging provider

Mapping of administrative console panel names to command names and IBM MQ names

Related information for WebSphere Application Server Version 7.0

Interoperation using the IBM MQ messaging provider

Managing messaging with the IBM MQ messaging provider

Mapping of administrative console panel names to command names and IBM MQ names

Configuring JMS resources using the administrative console

You can use the WebSphere Application Server administrative console to configure activation specifications, connection factories and destinations for the IBM MQ JMS provider.

About this task

You can use the WebSphere Application Server administrative console to create, view, or modify any of the following resources:

- Activation specifications
- Domain-independent connection factories (JMS 1.1 or later)
- Queue connection factories
- Topic connection factories
- Queues
- Topics

The following steps provide an overview of the ways in which you can use the administrative console to configure JMS resources for use with the IBM MQ messaging provider. Each step includes the name of the topic in the WebSphere Application Server product documentation to which you can refer for more information. See *Related links* for links to these topics in the WebSphere Application Server Version 8.5.5, Version 8.0 and Version 7.0 product documentation.

In a mixed-version WebSphere Application Server cell, you can administer IBM MQ resources on nodes of all versions. However, some properties are not available on all versions. In this situation, only the properties of that particular node are displayed in the administrative console.

Procedure

To create or configure an activation specification for use with the IBM MQ messaging provider:

- To create an activation specification, use the Create IBM MQ JMS resource wizard. You can either use the wizard to specify all the details for the activation specification, or you can choose to specify the

connection details for the IBM MQ by using a client channel definition table (CCDT). When you specify the connection details using the wizard, you can choose either to enter host and port information separately or, if you are using a multi-instance queue manager, to enter host and port information in the form of a connection name list. For more information, see *Creating an activation specification for the IBM MQ messaging provider*.
- To view or change the configuration properties of an activation specification, use the administrative console IBM MQ messaging provider connection factory settings panel. These configuration properties control how connections are created to associated queues and topics. For more information, see *Configuring an activation specification for the IBM MQ messaging provider*.

To create or configure a unified connection factory, a queue connection factory, or a topic connection factory for use with the IBM MQ messaging provider:
- To create a connection factory, first select the type of connection factory that you want to create, then use the Create IBM MQ JMS resource wizard to specify the details.
  - If your JMS application is intended to use only point-to-point messaging, create a domain-specific connection factory for the point-to-point messaging domain that can be used for creating connections specifically for point-to-point messaging.
  - If your JMS application is intended only to use publish/subscribe messaging, create a domain-specific connection factory for the publish/subscribe messaging domain that can be used for creating connections specifically for publish/subscribe messaging.
  - For JMS 1.1 or later, create a domain-independent connection factory that can be used for both point-to-point messaging and publish/subscribe messaging, allowing your application to perform both point-to-point and publish/subscribe work under the same transaction.

  You can choose whether to use the wizard to specify all the details for the connection factory, or you can choose to specify the connection details for the IBM MQ by using a client channel definition table (CCDT). When you specify the connection details using the wizard, you can choose either to enter host and port information separately or, if you are using a multi-instance queue manager, to enter host and port information in the form of a connection name list. For more information, see *Creating a connection factory for the IBM MQ messaging provider*.

To view or change the configuration properties of a connection factory:
- Use the administrative console connection factory settings panel for the type of connection factory that you want to configure. The configuration properties control how connections are created to associated queues and topics. For more information, see *Configuring a collection factory for the IBM MQ messaging provider*, or *Configuring a queue collection factory for the IBM MQ messaging provider*, or *Configuring a topic collection factory for the IBM MQ messaging provider*.

To configure a JMS queue destination for point-to-point messaging with the IBM MQ messaging provider:
- Use the administrative console IBM MQ messaging provider queue settings panel to define the following types of property:
  - General properties, including administration and IBM MQ queue properties.
  - Connection properties that specify how to connect to the queue manager that hosts the queue.
  - Advanced properties that control the behavior of connections made to IBM MQ messaging provider destinations.
  - Any custom properties for the queue destination.

  For more information, see *Configuring a queue for the IBM MQ messaging provider*.

To create or configure a JMS topic destination for publish/subscribe messaging with the IBM MQ messaging provider:
- Use the IBM MQ messaging provider topic settings panel to define the following types of property:
  - General properties, including administration and IBM MQ topic properties.
  - Advanced properties that control the behavior of connections made to IBM MQ messaging provider destinations.
  - Any custom properties for the queue destination.

  For more information, see *Configuring a topic for the IBM MQ messaging provider*.

Related concepts:

"Client channel definition table" on page 863
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 CCDT is created automatically. You must then make it available to the client application.

"Multi-instance queue managers" on page 1238
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 publish/subscribe messaging" on page 1165
You can start, stop and display the status of queued publish/subscribe. You can also add and remove streams, and add and delete queue managers from a broker hierarchy.

Related information for WebSphere Application Server traditional Version 9.0

IBM MQ messaging provider activation specifications

Creating an activation specification for the IBM MQ messaging provider

Configuring an activation specification for the IBM MQ messaging provider

Creating a connection factory for the IBM MQ messaging provider

Configuring a unified connection factory for the IBM MQ messaging provider

Configuring a queue connection factory for the IBM MQ messaging provider

Configuring a topic connection factory for the IBM MQ messaging provider

Configuring a queue for the IBM MQ messaging provider

Configuring a topic for the IBM MQ messaging provider

Related information for WebSphere Application Server Version 8.5.5

IBM WebSphere MQ messaging provider activation specifications

Creating an activation specification for the IBM WebSphere MQ messaging provider

Configuring an activation specification for the IBM WebSphere MQ messaging provider

Creating a connection factory for the IBM WebSphere MQ messaging provider

Configuring a unified connection factory for the IBM WebSphere MQ messaging provider

Configuring a queue connection factory for the IBM WebSphere MQ messaging provider

Configuring a topic connection factory for the IBM WebSphere MQ messaging provider

Configuring a queue for the IBM WebSphere MQ messaging provider

Configuring a topic for the IBM WebSphere MQ messaging provider

Related information for WebSphere Application Server Version 8.0

IBM WebSphere MQ messaging provider activation specifications

Creating an activation specification for the IBM WebSphere MQ messaging provider

Configuring an activation specification for the IBM WebSphere MQ messaging provider

Creating a connection factory for the IBM WebSphere MQ messaging provider

Configuring a unified connection factory for the IBM WebSphere MQ messaging provider

Configuring a queue connection factory for the IBM WebSphere MQ messaging provider

Configuring a topic connection factory for the IBM WebSphere MQ messaging provider

Configuring a queue for the IBM WebSphere MQ messaging provider

Configuring a topic for the IBM WebSphere MQ messaging provider

Related information for WebSphere Application Server Version 7.0

IBM WebSphere MQ messaging provider activation specifications

Creating an activation specification for the IBM WebSphere MQ messaging provider

Configuring an activation specification for the IBM WebSphere MQ messaging provider
Creating a connection factory for the IBM WebSphere MQ messaging provider
Configuring a unified connection factory for the IBM WebSphere MQ messaging provider
Configuring a queue connection factory for the IBM WebSphere MQ messaging provider
Configuring a topic connection factory for the IBM WebSphere MQ messaging provider
Configuring a queue for the IBM WebSphere MQ messaging provider
Configuring a topic for the IBM WebSphere MQ messaging provider

Configuring JMS resources using wsadmin scripting commands

You can use WebSphere Application Server wsadmin scripting commands to create, modify, delete or show information about JMS activation specifications, connection factories, queues and topics. You can also display and manage the settings for the IBM MQ resource adapter.

About this task

The following steps provide an overview of the ways in which you can use WebSphere Application Server wsadmin commands to configure JMS resources for use with the IBM MQ messaging provider. For more information about how to use these commands, see *Related links* for links to the WebSphere Application Server Version 8.5.5, Version 8.0 and Version 7.0 product documentation.

To run a command, use the AdminTask object of the wsadmin scripting client.

After using a command to create a new object or make changes, save your changes to the master configuration. For example, use the following command:

```
AdminConfig.save()
```

To see a list of the available IBM MQ messaging provider administrative commands, plus a brief description of each command, enter the following command at the wsadmin prompt:

```
print AdminTask.help('WMQAdminCommands')
```

To see overview help on a given command, enter the following command at the wsadmin prompt:

```
print AdminTask.help('command_name')
```

Procedure

To list all of the IBM MQ messaging provider resources defined at the scope at which a command is issued, use the following commands.

- To list the activation specifications, use the **`listWMQActivationSpecs`** command.
- To list the connection factories, use the **`listWMQConnectionFactories`** command.
- To list the queue type destinations, use the **`listWMQQueues`** command.
- To list the topic type destinations, use the **`listWMQTopics`** command.

To create a JMS resource for the IBM MQ messaging provider at a specific scope, use the following commands.

- To create an activation specification, use the **`createWMQActivationSpec`** command. You can either create an activation specification by specifying all the parameters to be used for establishing a connection, or you can create the activation specification so that it uses a client channel definition table (CCDT) to locate the queue manager to connect to.
- To create a connection factory, use the **`createWMQConnectionFactory`** command, using the **`-type`** parameter to specify the type of connection factory that you want to create:
  - If your JMS application is intended to use only point-to-point messaging, create a domain-specific connection factory for the point-to-point messaging domain that can be used for creating connections specifically for point-to-point messaging.

  - If your JMS application is intended only to use publish/subscribe messaging, create a domain-specific connection factory for the publish/subscribe messaging domain that can be used for creating connections specifically for publish/subscribe messaging.
  - For JMS 1.1 or later, create a domain-independent connection factory that can be used for both point-to-point messaging and publish/subscribe messaging, allowing your application to perform both point-to-point and publish/subscribe work under the same transaction.

  The default type is domain-independent connection factory.
- To create a queue type destination, use the **createWMQQueue** command.
- To create a topic type destination, use the **createWMQTopic** command.

To modify a JMS resource for the IBM MQ messaging provider at a specific scope, use the following commands.
- To modify an activation specification, use the **modifyWMQActivationSpec** command. You cannot change the type of an activation specification. For example, you cannot create an activation specification where you enter all the configuration information manually and then modify it to use a CCDT.
- To modify a connection factory, use the **modifyWMQConnectionFactory** command.
- To modify a queue type destination, use the **modifyWMQQueue** command.
- To modify a topic type destination, use the **modifyWMQTopic** command.

To delete a JMS resource for the IBM MQ messaging provider at a specific scope, use the following commands.
- To delete an activation specification, use the **deleteWMQActivationSpec** command.
- To delete a connection factory, use the **deleteWMQConnectionFactory** command.
- To delete a queue type destination, use the **deleteWMQQueue** command.
- To delete a topic type destination, use the **deleteWMQTopic** command.

To display information about a specific IBM MQ messaging provider resource, use the following commands.
- To display all the parameters, and their values, associated with a particular activation specification, use the **showWMQActivationSpec** command.
- To display all the parameters, and their values, associated with a particular connection factory, use the **showWMQConnectionFactory** command.
- To display all the parameters, and their values, associated with a particular queue type destination, use the **showWMQQueue** command.
- To display all the parameters, and their values, associated with a topic type destination, use the **deleteWMQTopic** command.

To manage settings for the IBM MQ resource adapter or the IBM MQ messaging provider, use the following commands.
- To manage the settings of the IBM MQ resource adapter that is installed at a particular scope, use the **manageWMQ** command.
- To display all the parameters, and their values that can be set by the **manageWMQ** command, use the **showWMQ** command. These settings are either related to the IBM MQ resource adapter or the IBM MQ messaging provider. The **showWMQ** command also shows any custom properties that are set on the IBM MQ resource adapter.

Related concepts:

"Client channel definition table" on page 863
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 CCDT is created automatically. You must then make it available to the client application.

"Multi-instance queue managers" on page 1238
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 publish/subscribe messaging" on page 1165
You can start, stop and display the status of queued publish/subscribe. You can also add and remove streams, and add and delete queue managers from a broker hierarchy.

Related information for WebSphere Application Server Version 8.5.5

createWMQActivationSpec command

createWMQConnectionFactory command

createWMQQueue command

createWMQTopic command

deleteWMQActivationSpec command

deleteWMQConnectionFactory command

deleteWMQQueue command

deleteWMQTopic command

listWMQActivationSpecs command

listWMQConnectionFactories command

listWMQQueues command

listWMQTopics command

modifyWMQActivationSpec command

modifyWMQConnectionFactory command

modifyWMQQueue command

modifyWMQTopic command

showWMQActivationSpec command

showWMQConnectionFactory command

showWMQQueue command

showWMQTopic command

showWMQ command

manageWMQ command

Related information for WebSphere Application Server Version 8.5.5

createWMQActivationSpec command

createWMQConnectionFactory command

createWMQQueue command

createWMQTopic command

deleteWMQActivationSpec command

deleteWMQConnectionFactory command

deleteWMQQueue command

deleteWMQTopic command

listWMQActivationSpecs command

listWMQConnectionFactories command

listWMQQueues command

listWMQTopics command

modifyWMQActivationSpec command

modifyWMQConnectionFactory command

modifyWMQQueue command

modifyWMQTopic command

showWMQActivationSpec command

showWMQConnectionFactory command

showWMQQueue command

showWMQTopic command

showWMQ command

manageWMQ command

Related information for WebSphere Application Server Version 8.0

createWMQActivationSpec command

createWMQConnectionFactory command

createWMQQueue command

createWMQTopic command

deleteWMQActivationSpec command

deleteWMQConnectionFactory command

deleteWMQQueue command

deleteWMQTopic command

listWMQActivationSpecs command

listWMQConnectionFactories command

listWMQQueues command

listWMQTopics command

modifyWMQActivationSpec command

modifyWMQConnectionFactory command

modifyWMQQueue command

modifyWMQTopic command

showWMQActivationSpec command

showWMQConnectionFactory command

showWMQQueue command

showWMQTopic command

showWMQ command

manageWMQ command

Related information for WebSphere Application Server Version 7.0

createWMQActivationSpec command

createWMQConnectionFactory command

createWMQQueue command

createWMQTopic command

deleteWMQActivationSpec command

deleteWMQConnectionFactory command

deleteWMQQueue command

deleteWMQTopic command

listWMQActivationSpecs command

listWMQConnectionFactories command

listWMQQueues command

listWMQTopics command

modifyWMQActivationSpec command

modifyWMQConnectionFactory command

modifyWMQQueue command

modifyWMQTopic command

showWMQActivationSpec command

showWMQConnectionFactory command

showWMQQueue command

showWMQTopic command

showWMQ command

manageWMQ command

Using JMS 2.0 shared subscriptions

In WebSphere Application Server traditional Version 9.0, you can configure and use JMS 2.0 shared subscriptions with IBM MQ Version 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 traditional Version 9.0 an IBM MQ Version 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 traditional administrative console.

Procedure

First create the objects in JNDI.

1. Create a topic destination in JNDI as normal (see “Configuring JMS resources using the administrative console” on page 1374).
2. Create the activation specification (see “Configuring JMS resources using the administrative console” on page 1374). 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 information:

Cloned and shared subscriptions

Subscription durability

Configuring the resource adapter for inbound communication

Related information for WebSphere Application Server traditional Version 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 traditional Version 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 traditional Version 9.0 an IBM MQ Version 9.0 resource adapter is pre-installed.

The following steps show how to customize and use these two properties using the WebSphere Application Server traditional administrative console.

Procedure

First create the objects in JNDI.

1. Create the ConnectionFactory in JNDI as normal (see "Configuring JMS resources using the administrative console" on page 1374).
2. Create the Destination in JNDI as normal (see "Configuring JMS resources using the administrative console" on page 1374). The Destination object must have the correct values.
3. Create the activation specification using any values that are needed (see "Configuring JMS resources using the administrative console" on page 1374). 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, or use annotations instead as described in step 5.

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" )}, mappedN
        @TransactionAttribute(TransactionAttributeType.REQUIRED)
        @TransactionManagement(TransactionManagementType.CONTAINER)
        publicclass LookupMDB implements MessageListener {
```

Related information:

Configuring the resource adapter for inbound communication

Related information for WebSphere Application Server traditional Version 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 the application server to use the latest resource adapter maintenance level

To ensure that the IBM MQ resource adapter is automatically updated to the latest available maintenance level when you apply WebSphere Application Server fix packs, you can configure all servers in your environment to use the latest version of the resource adapter contained in the WebSphere Application Server fix pack that you have applied to the installation of each node.

Before you begin

Important: If you are using WebSphere Application Server Version 7.0, Version 8 or Version 8.5 on any platform, do not install the IBM MQ Version 8.0 resource adapter into the application server. The IBM MQ Version 8.0 resource adapter can only be deployed into an application server that supports JMS 2.0. However, WebSphere Application Server Version 7.0, Version 8 and Version 8.5 only support JMS 1.1. These versions of WebSphere Application Server come with the IBM WebSphere MQ Version 7.0 resource adapter, which can be used to connect to a IBM MQ Version 8.0 queue manager using either the BINDINGS or CLIENT transport.

About this task

Use this task if any of the following circumstances apply to your configuration, and you want to configure all servers in your environment to use the latest version of the IBM MQ resource adapter:

- The JVM logs of any application server in your environment show the following IBM MQ resource adapter version information after WebSphere Application Server Version 7.0 Fix Pack 1 or later has been applied:
  ```
  WMSG1703I:RAR implementation Version 7.0.0.0-k700-L080820
  ```
- The JVM logs of any application server in your environment contain the following entry:
  ```
  WMSG1625E: It was not possible to detect
  the IBM MQ messaging provider code at the specified path <null>
  ```
- One or more nodes has previously been manually updated to use a specific maintenance level of the IBM MQ resource adapter that is now superseded by the latest version of the resource adapter contained in the current WebSphere Application Server maintenance level.

The *`profile_root`* directory that the examples refer to is the home directory for the WebSphere Application Server profile, for example `C:\Program Files\IBM\WebSphere\AppServer1`.

When you have performed the following steps for all cells and single server installations in your environment, your servers automatically receive maintenance to the IBM MQ resource adapter when a new WebSphere Application Server fix pack is applied.

Procedure

1. Start the application server. If the profile is part of a network deployment configuration, start the deployment manager and all node agents. If the profile contains an administrative agent, start the administrative agent.
2. Check the maintenance level of the IBM MQ resource adapter.
   a. Open a command prompt window and change to the *profile_root*\bin directory. For example, enter `cd C:\Program Files\IBM\WebSphere\AppServer1\bin`.
   b. Start the wsadmin tool by entering `wsadmin.bat -lang jython`, then if prompted to do so, enter your username and password.
   c. Type the following command, then press Return twice:

```
wmqInfoMBeansUnsplit = AdminControl.queryNames("WebSphere:type=WMQInfo,*")
wmqInfoMBeansSplit = AdminUtilities.convertToList(wmqInfoMBeansUnsplit)
for wmqInfoMBean in wmqInfoMBeansSplit: print wmqInfoMBean; print AdminControl.invoke(wmqInfoMBean, 'getInfo', '')
```

   You can also run this command in Jacl. For further information about how to do this, see *Ensuring that servers use the latest available IBM MQ resource adapter maintenance level* in the WebSphere Application Server product documentation.

   d. Find the `WMSG1703I` message in the displayed output from the command and check the resource adapter level. For example, for WebSphere Application Server Version 7.0 Fix Pack 15, the message should be:

```
WMSG1703I: RAR implementation Version 7.0.1.3-k701-103-100812
```

   This message shows that the version is 7.0.1.3-k701-103-100812, which is the correct resource adapter level for this 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 Fix Pack 15.

```
WMSG1703I: RAR implementation Version 7.0.0.0-k700-L080820
```

3. Copy the following Jython script into a file called `convertWMQRA.py`, then save it into the profile root directory, for example `C:\Program Files\IBM\WebSphere\AppServer1\bin`.

```
ras = AdminUtilities.convertToList(AdminConfig.list('J2CResourceAdapter'))

for ra in ras :
  desc = AdminConfig.showAttribute(ra, "description")
  if (desc == "WAS 7.0 Built In IBM MQ Resource Adapter") or (desc == "WAS 7.0.0.1 Built In IBM MQ Resource Adapter"):
    print "Updating archivePath and classpath of " + ra
    AdminConfig.modify(ra, [['archivePath', "${WAS_INSTALL_ROOT}/installedConnectors/wmq.jmsra.rar"]])
    AdminConfig.unsetAttributes(ra, ['classpath'])
    AdminConfig.modify(ra, [['classpath', "${WAS_INSTALL_ROOT}/installedConnectors/wmq.jmsra.rar"]])
    AdminConfig.save()
  #end if
#end for
```

   Tip: When saving the file, make sure that it is saved as a python file rather than a text file.

4. Use the WebSphere Application Server wsadmin tool to run the Jython script that you have just created. Open a command prompt and navigate to the `\bin` directory in the home directory for the WebSphere Application Server, for example `C:\Program Files\IBM\WebSphere\AppServer1\bin` directory, then type the following command and press Return:

```
wsadmin -lang jython -f convertWMQRA.py
```

   If prompted to do so, enter your username and password.

   Note: If you run the script against a profile that is part of a network deployment configuration, the script updates all profiles that need updating in that configuration. A full resynchronization might be necessary if you have pre-existing configuration file inconsistencies.

5. If you are running in a network deployment configuration, ensure that the node agents are fully re-synchronized. For more information, see Synchronizing nodes using the wsadmin scripting tool or Adding, managing, and removing nodes.

6. Stop all servers in the profile. If the profile is part of a network deployment configuration, also stop any cluster members in the configuration, stop all node agents in the configuration, and stop the deployment manager. If the profile contains an administrative agent, stop the administrative agent.
7. Run the **`osgiCfgInit`** command from the *`profile_root`*`/bin` directory. The `osgiCfgInit` command resets the class cache used by the OSGi runtime environment. If the profile is part of a network deployment configuration, run the **`osgiCfgInit`** command from the `profile_root/bin` directory of every profile that is part of the configuration.
8. Restart all servers in the profile. If the profile is part of a network deployment configuration, also restart any cluster members in the configuration, restart all node agents in the configuration, and restart the deployment manager. If the profile contains an administrative agent, restart the administrative agent.
9. Repeat step 2 to check that the resource adapter is now at the correct level.

What to do next

If you continue to experience problems after performing the steps described in this topic, and you have previously used the **Update resource adapter** button on the JMS Provider Settings panel in the WebSphere Application Server administrative console to update the IBM MQ resource adapter on any nodes in your environment, it is possible that you are experiencing the issue described in APAR PM10308.

Related information:

Using the IBM MQ resource adapter

Related information for WebSphere Application Server Version 8.5.5

Ensuring that servers use the latest available IBM MQ resource adapter maintenance level

Synchronizing nodes using the wsadmin scripting tool

Adding, managing, and removing nodes

JMS provider settings

Related information for WebSphere Application Server Version 8.0

Ensuring that servers use the latest available IBM MQ resource adapter maintenance level

Synchronizing nodes using the wsadmin scripting tool

Adding, managing, and removing nodes

JMS provider settings

Related information for WebSphere Application Server Version 7.0

Ensuring that servers use the latest available IBM MQ resource adapter maintenance level

Synchronizing nodes using the wsadmin scripting tool

Adding, managing, and removing nodes

JMS provider settings

Configuring the JMS `PROVIDERVERSION` property

The IBM MQ messaging provider has three modes of operation: normal mode, normal mode with restrictions, and migration mode. You can set the JMS **`PROVIDERVERSION`** property to select which of these modes a JMS application uses to publish and subscribe.

About this task

The selection of the IBM MQ messaging provider mode of operation can be primarily controlled by setting the PROVIDERVERSION connection factory property. The mode of operation can also be selected automatically if a mode has not been specified.

The **PROVIDERVERSION** property differentiates between the three IBM MQ messaging provider modes of operation:

IBM MQ messaging provider normal mode
Normal mode uses all the features of an IBM MQ queue manager to implement JMS. This mode is optimized to use the JMS 2.0 API and functionality.

IBM MQ messaging provider normal mode with restrictions
Normal mode with restrictions uses the JMS 2.0 API, but not the new features, that is, shared subscriptions, delayed delivery, and asynchronous send.

IBM MQ messaging provider migration mode
With migration mode, you can connect to a IBM MQ Version 8.0 or later queue manager, but none of the features of a IBM WebSphere MQ Version 7.0 or later queue manager, such as read ahead and streaming, are used.

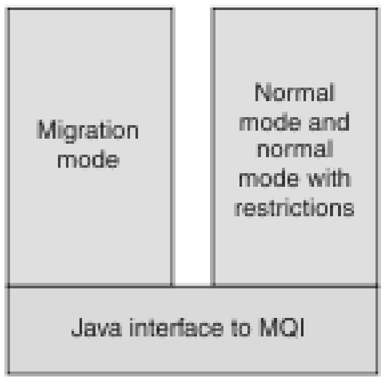


Figure 168. Messaging provider modes

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, seeCreating and configuring connection factories and destinations in an IBM MQ classes for JMS application.

To override connection factory provider mode settings for all connection factories in the JVM:

- To override connection factory provider mode settings, use the `com.ibm.msg.client.wmq.overrideProviderVersion` property If you cannot change the connection factory that you are using, you can use the `com.ibm.msg.client.wmq.overrideProviderVersion` property to override any setting on the connection factory. This override applies to all connection factories in the JVM but the actual connection factory objects are not modified.

Related information:
PROVIDERVERSION
JMS provider version troubleshooting
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 Version 6.0.

`unspecified` (the default value)
: The JMS client uses an algorithm to determine which mode of operation is used.

The value that you specify for the **`PROVIDERVERSION`** property must be a string. If you are specifying an option of 8, 7 or 6, you can do this in any of the following formats:
- V.R.M.F
- V.R.M
- V.R
- V

where V, R, M and F are integer values greater than or equal to zero. The extra R, M and F values are optional and are available for you to use in case fine grained control is needed. For example, if you wanted to use a **`PROVIDERVERSION`** level of 7, you could set **`PROVIDERVERSION`** = `7`, `7.0`, `7.0.0` or `7.0.0.0`.

Types of connection factory object

You can set the **`PROVIDERVERSION`** property for the following types of connection factory object:
- MQConnectionFactory
- MQQueueConnectionFactory
- MQTopicConnectionFactory
- MQXAConnectionFactory
- MQXAQueueConnectionFactory
- MQXAQueueConnectionFactory
- MQXAQueueConnectionFactory
- MQXATopicConnectionFactory

For more information about these different types of connection factory, see “Configuring JMS objects using the administration tool” on page 1365.

Related information:

Architecture and overview of features

`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.

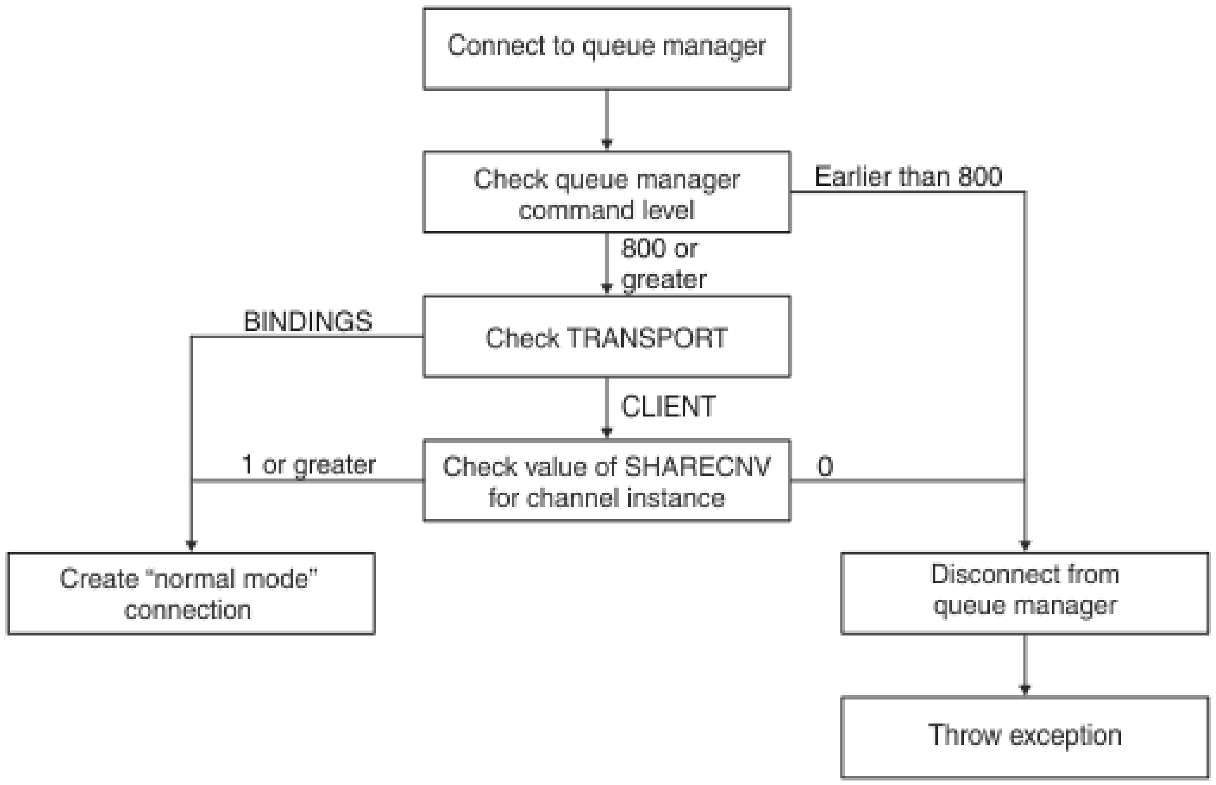

Connect to queue manager
Check queue manager command level
Earlier than 800
800 or greater
BINDINGS
Check TRANSPORT
CLIENT
1 or greater
Check value of SHARECNV for channel instance
0
Create "normal mode" connection
Disconnect from queue manager
Throw exception

Figure 169. PROVIDERVERSION normal mode

If the queue manager specified in the connection factory settings has a command level of 800 or greater, and the **`TRANSPORT`** property of the connection factory is set to BINDINGS, a normal mode connection is created without checking any further properties.

If the queue manager specified in the connection factory settings has a command level of 800 or greater, and the **`TRANSPORT`** property is set to CLIENT, the **`SHARECNV`** property on the server connection channel is also checked. This check is needed because IBM MQ messaging provider normal mode uses the sharing conversations feature. Therefore, for a normal mode connection attempt to be successful, the **`SHARECNV`** property, which controls the number of conversations that can be shared, must have a value of 1 or greater.

If all the checks shown in the flowchart are successful, a normal mode connection to the queue manager is created and all of the JMS 2.0 API and features, that is, asynchronous send, delayed delivery, and shared subscription, can then be used.

An attempt to create a normal mode connection fails for either of the following reasons:

- The queue manager specified in the connection factory settings has a command level that is earlier than 800. In this case, the createConnection method fails with an exception `JMSFMQ0003`.
- The **`SHARECNV`** property on the server connection channel is set to `0`. If this property does not have a value of 1 or greater, the createConnection method fails with an exception `JMSCC5007`.

Related information:

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 Version 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 .

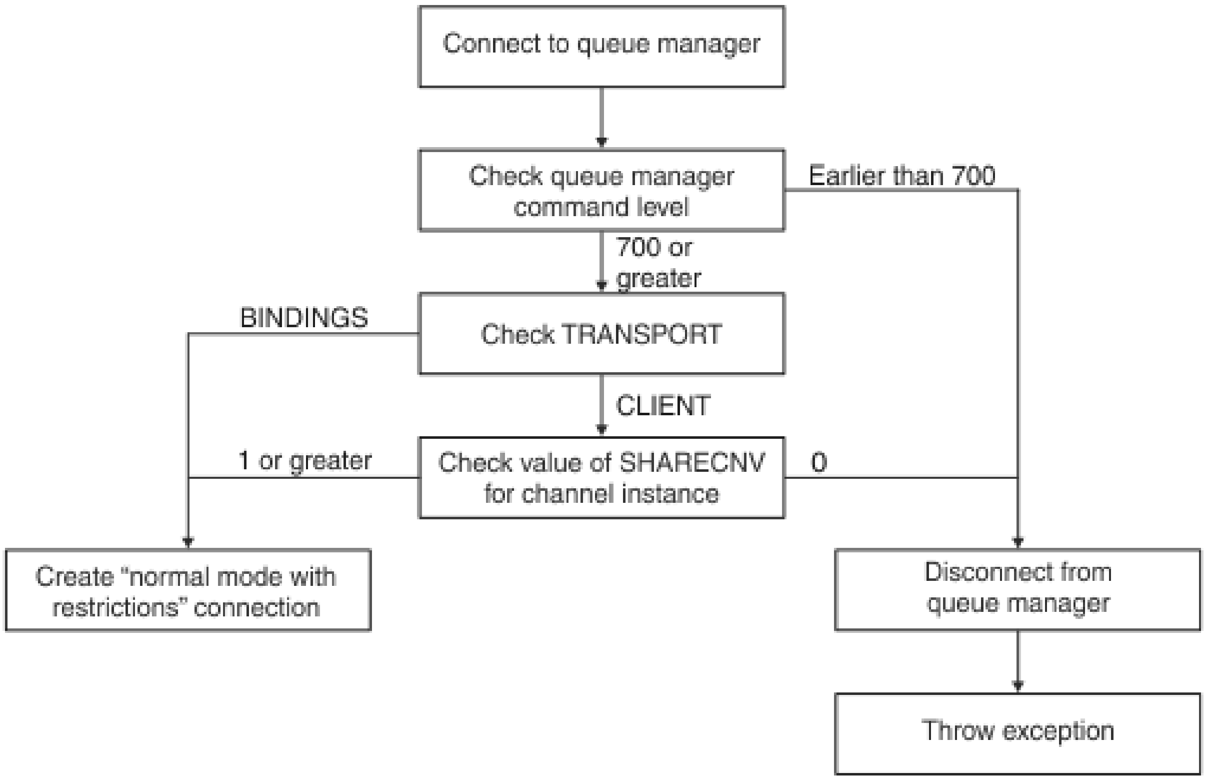

Connect to queue manager
Check queue manager command level
Earlier than 700
700 or greater
BINDINGS
Check TRANSPORT
CLIENT
1 or greater
Check value of SHARECNV for channel instance
0
Create "normal mode with restrictions" connection
Disconnect from queue manager
Throw exception

Figure 170. PROVIDERVERSION normal mode with restrictions

If the queue manager specified in the connection factory settings has a command level of 700 or greater, and the **TRANSPORT** property of the connection factory is set to BINDINGS, a normal mode connection is created without checking any further properties.

If the queue manager specified in the connection factory settings has a command level of 700 or greater, and the **TRANSPORT** property is set to CLIENT, the **SHARECNV** property on the server connection channel is also checked. This check is needed because IBM MQ messaging provider normal mode with restrictions uses the sharing conversations feature. Therefore, for a normal mode with restrictions connection attempt to be successful, the **SHARECNV** property, which controls the number of conversations that can be shared, must have a value of 1 or greater.

If all the checks shown in the flowchart are successful, a normal mode with restrictions connection to the queue manager is created and you can then use the JMS 2.0 API, but not the asynchronous send, delayed delivery, or shared subscription features.

An attempt to create a normal mode with restrictions connection fails for either of the following reasons:
- The queue manager specified in the connection factory settings has a command level that is earlier than 700. In this case, the createConnection method fails with exception `JMSFCC5008`.
- The **SHARECNV** property on the server connection channel is set to `0`. If this property does not have a value of 1 or greater, the createConnection method fails with an exception `JMSCC5007`.

Related information:

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 Version 6.0, such as queued publish/subscribe, selection implemented on the client side, non-multiplex channels, and polling used to implement listeners.

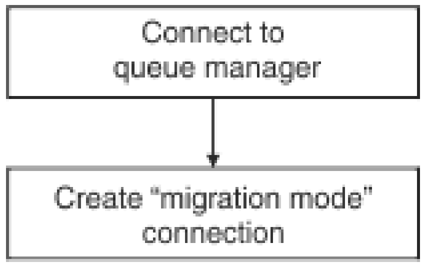


Figure 171. PROVIDERVERSION migration mode

If you want to connect to WebSphere Message Broker Version 6.0 or 6.1 using IBM MQ Enterprise Transport Version 6.0, you must use migration mode.

You can connect to an IBM MQ Version 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 Version 8.0 or later client connecting to an IBM MQ Version 8.0 or later queue manager on a distributed platform, z/OS or an IBM MQ Version 8.0 or later queue manager on z/OS, then the message selection is done by the queue manager rather than on the client system.

If IBM MQ messaging provider migration mode is specified and the IBM MQ classes for JMS attempt to use any of the JMS 2.0 API, the API method call fails with the exception JMSCC5007.

Related information:

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.

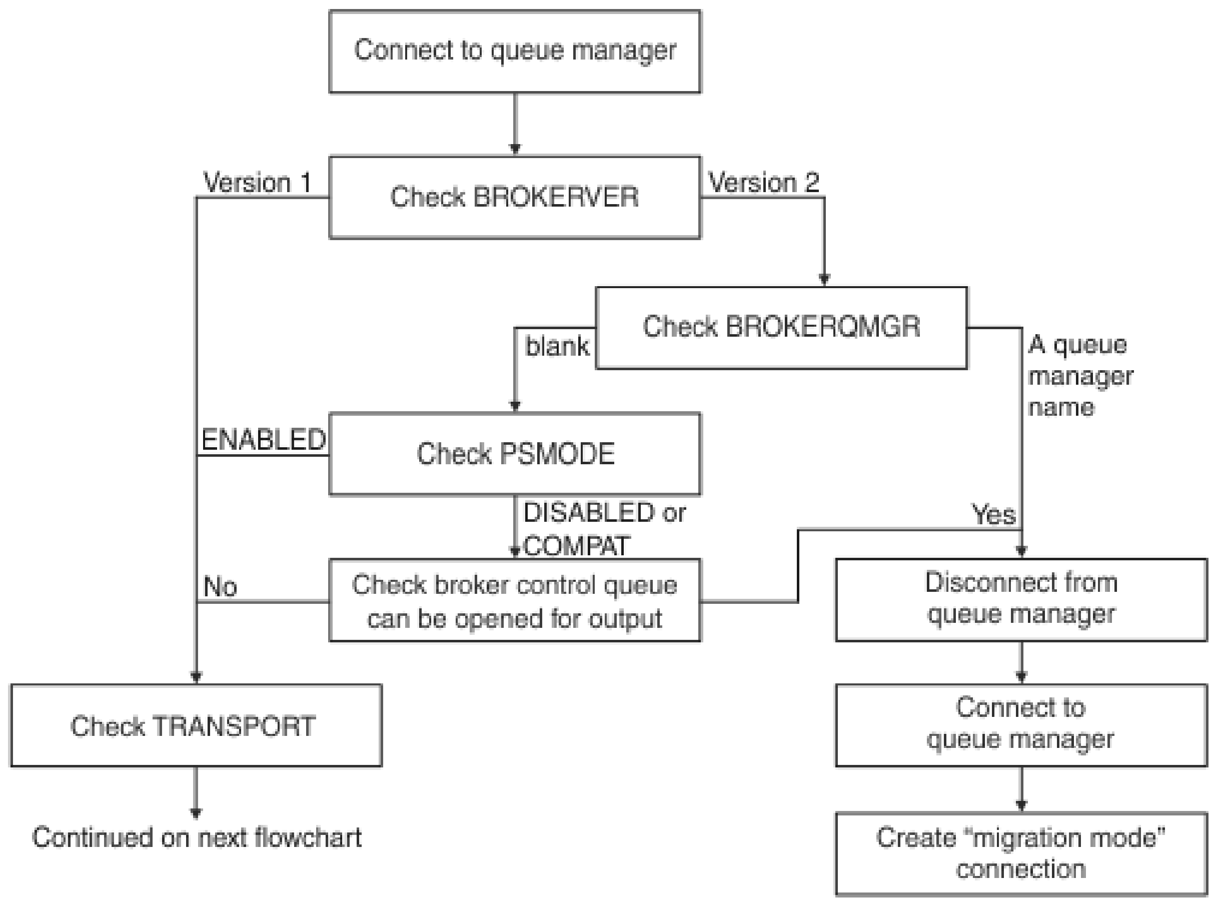


Figure 172. PROVIDERVERSION unspecified

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.

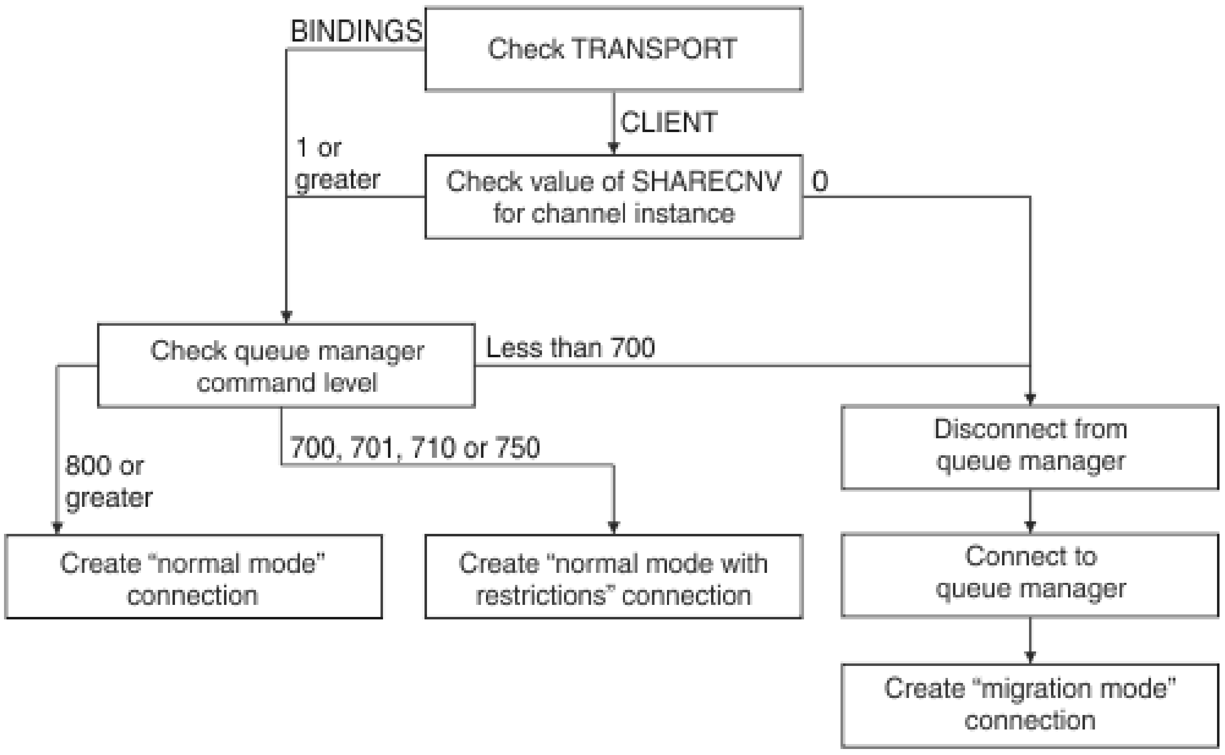
If the property values are set as required for a normal mode connection, checking next moves on to the **TRANSPORT** property as shown in Figure 2.

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.



BINDINGS
Check TRANSPORT
CLIENT
1 or greater
Check value of SHARECNV for channel instance
0
Check queue manager command level
Less than 700
800 or greater
700, 701, 710 or 750
Disconnect from queue manager
Connect to queue manager
Create "normal mode" connection
Create "normal mode with restrictions" connection
Create "migration mode" connection

Figure 173. PROVIDERVERSION unspecified (continued)

A normal mode connection is created in either of the following cases:

- The **TRANSPORT** property of the connection factory is set to BINDINGS, and the queue manager has a command level of 800 or greater.
- The **TRANSPORT** property is set to CLIENT, the **SHARECNV** property on the server connection channel has a value of 1 or greater, and the queue manager has a command level of 800 or greater.

If the queue manager has a command level of 700, 701, 710 or 750, a normal mode with restrictions connection to the queue manager is created.

If the queue manager has a command level of less than 700, the JMS client disconnects from the queue manager and then connects with the queue manager again to create a migration mode connection.

A migration mode connection is also created if the **TRANSPORT** property is set to CLIENT and the **SHARECNV** property on the server connection channel has a value of 0.

Related information:

Dependencies between properties of IBM MQ classes for JMS objects

ALTER QMGR (PSMODE attribute)

BROKERCONQ

BROKERQMGR

BROKERVER

DEFINE CHANNEL (SHARECNV property)

TRANSPORT

When to override the `PROVIDERVERSION` default setting

If a connection factory that was created in the JNDI namespace with a previous version of IBM MQ classes for JMS is used with the new version of IBM MQ classes for JMS, the **PROVIDERVERSION** property for the connection factory is set to the default value of `unspecified` and an algorithm is used to determine which IBM MQ messaging provider mode of operation is used. However, there are two cases where you must override the default selection for the **PROVIDERVERSION** property so that the IBM MQ classes for JMS can work correctly.

Note: The migration mode that is described in this topic is for migration from IBM WebSphere MQ Version 6.0 to IBM WebSphere MQ Version 7.0. It does not apply to migration from later releases.

IBM WebSphere MQ Version 6.0, WebSphere Application Server Version 6.0.x, and WebSphere Message Broker Version 6 are out of support, and therefore this topic is included only for reference purposes.

When the **PROVIDERVERSION** property is set to the default of `unspecified`, an algorithm is used to determine which mode of operation to use, as described in "**PROVIDERVERSION** unspecified" on page 1393. However, you cannot use this algorithm in the following two scenarios.

1. If WebSphere Message Broker and WebSphere Event Broker are in compatibility mode, you must specify a value for the **PROVIDERVERSION** property for WebSphere Message Broker and WebSphere Event Broker to work correctly.
2. If you are using WebSphere Application Server Version 6.0.1, WebSphere Application Server Version 6.0.2, or WebSphere Application Server Version 6.1, connection factories are defined by using the WebSphere Application Server administrative console.

   In WebSphere Application Server, the default value of the **BROKERVER** property on a connection factory is `V2`. The default value for the **BROKERVER** property for connection factories that are created by using the JMS administration tool **`JMSAdmin`** or IBM MQ Explorer is `V1`. This property is now `unspecified` in IBM MQ.

If the **BROKERVER** property is set to `V2`, either because it was created by WebSphere Application Server or the connection factory has been used for publish/subscribe before, and has an existing queue manager that has a **BROKERCONQ** property defined (because it has been used for publish/subscribe messaging before), the IBM MQ messaging provider migration mode is used.

However, if you want the application to use peer-to-peer communication and the application is using an existing queue manager that has ever been used for publish/subscribe, and has a connection factory with **BROKERVER** set to `2`, which is the default setting if the connection factory was created in WebSphere Application Server, the IBM MQ messaging provider migration mode is used. Using IBM MQ messaging provider migration mode in this case is unnecessary; use IBM MQ messaging provider normal mode instead. You can use one of the following methods to work around this:

- Set **BROKERVER** to `1` or unspecified. The option that you choose depends on your application.
- Set **PROVIDERVERSION** to `8`, or `7`, which are custom properties in WebSphere Application Server Version 6.1.

  Alternatively, use the client configuration property, or modify the queue manager connected so that it does not have the **BROKERCONQ** property set, or make the queue unusable.

Configuring provider version information in WebSphere Application Server

To configure provider version information in WebSphere Application Server, you can either use the administrative console or wsadmin commands.

Procedure

To configure provider version information for an IBM MQ connection factory or activation specification object in WebSphere Application Server, see the *Related information* for links to further information in the WebSphere Application Server product documentation.

Related information for WebSphere Application Server Version 8.5.5

IBM MQ messaging provider connection factory settings

`createWMQConnectionFactory` command

IBM MQ messaging provider activation specification settings

`createWMQActivationSpec` command

Related information for WebSphere Application Server Version 8.0.0

IBM MQ messaging provider connection factory settings

`createWMQConnectionFactory` command

IBM MQ activation specification settings

`createWMQActivationSpec` command

Related information for WebSphere Application Server Version 7.0.0

IBM MQ messaging provider connection factory settings

`createWMQConnectionFactory` command

IBM MQ activation specification settings

`createWMQActivationSpec` command

Configuring the IBM MQ Console and REST API

V 9.0.1

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

- “Configuring security” on page 1398
- “Configuring the HTTP host name” on page 1399
- “Configuring the HTTP and HTTPS ports” on page 1401
- “Configuring the response timeout” on page 1402
- “Configuring autostart” on page 1403
- “Configuring logging” on page 1405
- “Configuring the LTPA token expiry interval” on page 1407
- “Configuring the messaging REST API” on page 1409
- “Configuring CSRF protection” on page 1398

Configuring security

V 9.0.1

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 Application Server Liberty. You can then authorize those users by assigning users and groups a role. At Version 9.0.1, there is no security for the REST API. From Version 9.0.2, you can configure security for the REST API.

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. 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. This token enables the user to use the IBM MQ Console without re-authenticating until the token expires. You can configure when the token expires. For more information, see “Configuring the LTPA token expiry interval” on page 1407.

Procedure

- IBM MQ Console and REST API security
- “Configuring the LTPA token expiry interval” on page 1407

Configuring CSRF protection

V 9.0.4

Cross-Site Request Forgery (CSRF) is a type of attack that occurs when a malicious website causes a user's browser to perform an unwanted action on a trusted site for which the user is currently authenticated. .

Before you begin

You must be a privileged user to complete this procedure.

V 9.0.4 You can view the current configuration of the CSRF protection by using the following command:

```
dspmqweb properties -a
```

The `mqRestCsrfValidation` field shows whether CSRF validation checks are performed. For more information, see dspmqweb.

Note: V 9.0.5 The `mqRestCsrfExpirationInMinutes` field, introduced in Version 9.0.4 to show the CSRF expiration time, no longer exists in Version 9.0.5.

Attention: z/OS V 9.0.4

Before issuing 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 do this, 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.sh`. For example:

```
export WLP_USER_DIR=/var/mqm/web/installation1
```

For more information, see Create the Liberty server definition.

About this task

V 9.0.5 Prior to Version 9.0.5 the IBM MQ Console and REST API use a synchronizer token to protect against CSRF attacks. In Version 9.0.4 only, CSRF synchronizer tokens for the administrative REST API are periodically regenerated. From Version 9.0.5, CSRF synchronizer tokens are not used. Instead, a custom HTTP header needs to be set, which provides equivalent protection to using a synchronizer token.

You can modify configuration of the CSRF protection for the REST API by using the **`setmqweb properties`** command

Procedure

Use the following method to configure CSRF token validation for the REST API:

- For Version 9.0.4 only, use the **`setmqweb properties`** command to alter token expiry:

  ```
  setmqweb properties -k mqRestCsrfExpirationInMinutes -v time
  ```

  where *time* specifies the time, in minutes, before the CSRF token expires. The token remains valid for the next HTTP POST, PATCH, or DELETE method after its expiration, after which, a new token is returned as a cookie and the previous token value is invalidated. A time value of -1 disables CSRF token expiration, while a value of 0 causes the token to be changed on every POST, PATCH or DELETE request. The default value is 30 minutes.
- Use the **`setmqweb properties`** command to remove CSRF validation checks:

  ```
  setmqweb properties -k mqRestCsrfValidation -v boolean
  ```

  where *boolean* specifies whether CSRF validation checks are performed, a value of false removes CSRF token validation checks. Validation of tokens is recommended, particularly where users are using web browsers to access the REST API. The default value is true, and CSRF tokens are validated for all HTTP POST, PATCH, and DELETE requests via the REST API.

Configuring the HTTP host name

V 9.0.1

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. V 9.0.4 From Version 9.0.4, you can configure host name to allow remote connections by using the **`setmqweb`** command. In IBM MQ Version 9.0.3, and earlier, you can configure host name to allow remote connections by editing the `mqwebuser.xml` file.

Before you begin

You must be a privileged user to complete this procedure.

V 9.0.4 From Version 9.0.4,you can 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. For more information, see dspmqweb.

Attention: z/OS V 9.0.4

Before issuing 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 do this, 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.sh`. For example:

```
export WLP_USER_DIR=/var/mqm/web/installation1
```

For more information, see Create the Liberty server definition.

Procedure

V 9.0.4 Use one of the following methods to configure the host name:

- From Version 9.0.4, use the **`setmqweb properties`** 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.
- For Version 9.0.3 and earlier, edit the `mqwebuser.xml` file:
  1. Open the `mqwebuser.xml` file.

     The `mqwebuser.xml` file can be found in one of the following directories:

     – ULW On UNIX, Linux, and Windows: *`MQ_DATA_DIRECTORY`*`/web/installations/`*`installationName`*`/servers/mqweb`

     – z/OS On z/OS: *`WLP_user_directory`*`/servers/mqweb`

       where *WLP_user_directory* is the directory that was specified when the **`crtmqweb.sh`** script ran to create the mqweb server definition.
  2. Configure the mqweb server:

     – To allow remote connections to the mqweb server, add the following line to the `mqwebuser.xml` file, within the `<server>` tags:

       ```
       <variable name="httpHost" value="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 (*) to specify all available network interfaces.

     – To allow only local connections to the mqweb server, either remove the following line from the `mqwebuser.xml` file, or set the value to `localhost`:

       ```
       <variable name="httpHost" value="hostName" />
       ```

Configuring the HTTP and HTTPS ports

V 9.0.1

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. V 9.0.4 From Version 9.0.4, you can configure the ports by using the **setmqweb** command. In IBM MQ Version 9.0.3, and earlier, you can configure the ports by editing the `mqwebuser.xml` file.

Before you begin

You must be a privileged user to complete this procedure.

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 from a browser. You can configure the mqweb server to prevent this behavior, and make the environment more secure, by adding the following line to the `mqwebuser.xml` file:

```
<webAppSecurity ssoRequiresSSL="true"/>
```

V 9.0.4 From Version 9.0.4,you can 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. For more information, see dspmqweb.

Attention: z/OS V 9.0.4

Before issuing 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 do this, 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.sh`. For example:

```
export WLP_USER_DIR=/var/mqm/web/installation1
```

For more information, see Create the Liberty server definition.

Procedure

V 9.0.4 Use one of the following methods to configure the ports:

- From Version 9.0.4, use the **setmqweb properties** command:
  - To enable or configure the HTTP port, use 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.
  - To configure the HTTPS port, use 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.
- For Version 9.0.3 and earlier, edit the `mqwebuser.xml` file:

1. Open the mqwebuser.xml file.

   The mqwebuser.xml file can be found in one of the following directories:

   - ULW On UNIX, Linux, and Windows: *MQ_DATA_DIRECTORY*/web/installations/*installationName*/servers/mqweb
   - z/OS On z/OS: *WLP_user_directory*/servers/mqweb

     where *WLP_user_directory* is the directory that was specified when the **crtmqweb.sh** script ran to create the mqweb server definition.
2. Configure the ports:
   - To enable or configure the HTTP port, add or edit the following line in the mqwebuser.xml file, within the <server> tags:

     ```
     <variable name="httpPort" value="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.
   - To configure the HTTPS port, add or edit the following line in the mqwebuser.xml file, within the <server> tags:

     ```
     <variable name="httpsPort" value="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.

Configuring the response timeout

V 9.0.1

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. V 9.0.4 From Version 9.0.4, you can configure the IBM MQ Console and REST API to use a different timeout value by using the **setmqweb** command. In IBM MQ Version 9.0.3, and earlier, you can configure the IBM MQ Console and REST API to use a different timeout value by editing the mqwebuser.xml file.

Before you begin

You must be a privileged user to complete this procedure.

V 9.0.4 From Version 9.0.4,you can view the current configuration of the REST API response 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.

Attention: z/OS V 9.0.4

Before issuing 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 do this, 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.sh`. For example:

```
export WLP_USER_DIR=/var/mqm/web/installation1
```

For more information, see Create the Liberty server definition.

Procedure

V 9.0.4 Use one of the following methods to configure the timeout:

- From Version 9.0.4, use the **setmqweb properties** command:

  ```
  setmqweb properties -k mqRestRequestTimeout -v timeout
  ```

  where *timeout* specifies the time, in seconds, before the time out.
- For Version 9.0.3 and earlier, edit the `mqwebuser.xml` file:
  1. Open the `mqwebuser.xml` file.

     The `mqwebuser.xml` file can be found in one of the following directories:
     - ULW On UNIX, Linux, and Windows: `MQ_DATA_DIRECTORY/web/installations/installationName/servers/mqweb`
     - z/OS On z/OS: `WLP_user_directory/servers/mqweb`

       where *WLP_user_directory* is the directory that was specified when the **crtmqweb.sh** script ran to create the mqweb server definition.
  2. Configure the timeout by adding or editing the following line in the `mqwebuser.xml` file, within the `<server>` tags:

     ```
     <variable name="mqRestRequestTimeout" value="timeout" />
     ```

     where *timeout* specifies the time, in seconds, before the time out.

Configuring autostart

V 9.0.1

By default, the IBM MQ Console is automatically started when the mqweb server starts. In Version 9.0.1, the REST API is not automatically started. From Version 9.0.2, the REST API is automatically started when the mqweb server starts. V 9.0.4 From Version 9.0.4, you can configure whether the IBM MQ Console and the REST API start automatically by using the **setmqweb** command. In IBM MQ Version 9.0.3, and earlier, you can configure whether the IBM MQ Console and the REST API start automatically by editing the `mqwebuser.xml` file.

Before you begin

You must be a privileged user to complete this procedure.

V 9.0.4 From Version 9.0.4,you can view the current configuration of the REST API 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. For more information, see dspmqweb.

Attention: z/OS V 9.0.4

Before issuing 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 do this, 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.sh`. For example:

```
export WLP_USER_DIR=/var/mqm/web/installation1
```

For more information, see Create the Liberty server definition.

Procedure

V 9.0.4 Use one of the following methods to configure whether the IBM MQ Console and the REST API start automatically:

- From Version 9.0.4, use the **`setmqweb properties`** command:
  - Configure whether the IBM MQ Console automatically starts, 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 requires a manual start, 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.
- For Version 9.0.3 and earlier, edit the `mqwebuser.xml` file:
  1. Open the `mqwebuser.xml` file.

     The `mqwebuser.xml` file can be found in one of the following directories:
     - ULW On UNIX, Linux, and Windows: *`MQ_DATA_DIRECTORY`*`/web/installations/`*`installationName`*`/servers/mqweb`
     - z/OS On z/OS: *`WLP_user_directory`*`/servers/mqweb`

       where *WLP_user_directory* is the directory that was specified when the **`crtmqweb.sh`** script ran to create the mqweb server definition.
  2. Configure autostart:
     - Configure whether the IBM MQ Console requires a manual start by adding or updating the following line in the `mqwebuser.xml` file, within the <server> tags:

       ```
       <variable name="mqConsoleAutostart" value="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 requires a manual start by adding or updating the following line in the `mqwebuser.xml` file, within the <server> tags:

       ```
       <variable name="mqRestAutostart" value="start" />
       ```

       where *start* is the value `True` if you want the REST API to automatically start, or `False` otherwise.

Configuring logging

V 9.0.1

You can configure the logging levels, maximum log file size, and the maximum number of log files that are used by the mqweb server which hosts the IBM MQ Console and REST API. V 9.0.4 From Version 9.0.4, you can configure logging by using the **setmqweb** command. In IBM MQ Version 9.0.3, and earlier, you can configure logging by editing the `mqwebuser.xml` file.

Before you begin

You must be a privileged user to complete this procedure.

V 9.0.4 From Version 9.0.4,you can view the current configuration of the REST API logging by using the following command:

```
dspmqweb properties -a
```

The `maxTraceFileSize` field shows the maximum trace file size, the `maxTraceFiles` field shows the maximum number of trace files, and the `traceSpec` field shows the level of trace used. For more information, see dspmqweb.

Attention: z/OS V 9.0.4

Before issuing 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 do this, 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.sh`. For example:

```
export WLP_USER_DIR=/var/mqm/web/installation1
```

For more information, see Create the Liberty server definition.

About this task

The log files for the mqweb server can be found in one of the following directories:

- ULW On UNIX, Linux, and Windows: *`MQ_DATA_DIRECTORY`*`/web/installations/`*`installationName`*`/servers/mqweb/logs`
- z/OS On z/OS: *`WLP_user_directory`*`/servers/mqweb/logs`

  where *WLP_user_directory* is the directory that was specified when the **crtmqweb.sh** script ran to create the mqweb server definition.

For more information about enabling trace for the IBM MQ Console and REST API, see Tracing the IBM MQ Console and REST API.

Procedure

V 9.0.4 Use one of the following methods to configure logging:

- From Version 9.0.4, use the **setmqweb properties** command:
  - To set the maximum log file size, use the following command:

    ```
    setmqweb properties -k maxTraceFileSize -v size
    ```

where *size* specifies the size, in MB, that each log file can reach. The default value is 20.

– To set the maximum number of files to use for logging, use the following command:

```
setmqweb properties -k maxTraceFiles -v max
```

where *max* specifies the maximum number of files. The default value is 2.

– To configure the level of logging that is used, use the following command:

```
setmqweb properties -k traceSpec -v level
```

where *level* is one of the values listed in Table 135. The table outlines the logging levels in increase 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.

The default value is **`*=info`**. Change this value when IBM Service requests it.

Table 135. 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

• For Version 9.0.3 and earlier, edit the `mqwebuser.xml` file:

1. Open the `mqwebuser.xml` file.

   The `mqwebuser.xml` file can be found in one of the following directories:

   – ULW On UNIX, Linux, and Windows: *`MQ_DATA_DIRECTORY`*`/web/installations/`*`installationName`*`/servers/mqweb`

   – z/OS On z/OS: *`WLP_user_directory`*`/servers/mqweb`

   where *WLP_user_directory* is the directory that was specified when the **`crtmqweb.sh`** script ran to create the mqweb server definition.

2. Configure logging:

   – To set the maximum log file size, add or edit the following line in the `mqwebuser.xml` file, within the `<server>` tags:

   ```
   <variable name="maxTraceFileSize" value="size" />
   ```

   where *size* specifies the size, in MB, that each log file can reach. The default value is 20.

   – To set the maximum number of files to use for logging, add or edit the following line in the `mqwebuser.xml` file, within the `<server>` tags:

```
<variable name="maxTraceFiles" value="max" />
```

where *max* specifies the maximum number of files. The default value is 2.

- To configure the level of logging that is used, add or edit the following line in the `mqwebuser.xml` file, within the `<server>` tags:

  ```
  <variable name="traceSpec" value="level" />
  ```

  where *level* is one of the values listed in the Table 135 on page 1406 table.

  The table outlines the logging levels in increase 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.

  The default value is **`*=info`**. Change this value when IBM Service requests it.

Configuring the LTPA token expiry interval

V 9.0.1

LTPA tokens can be used to avoid needing a user to provide username and password credentials on each request to WebSphere Application Server Liberty. You can configure the expiry interval for LTPA authentication tokens.

Before you begin

You must be a privileged user to complete this procedure.

V 9.0.4 From Version 9.0.4, you can view the current configuration of the token expiry by using the **dspmqweb properties** command with the `-a` flag. For more information, see dspmqweb. You can reset the value of the token expiry by using the **setmqweb properties** command with the `-k` and `-d` flags. For more information, see setmqweb.

V 9.0.2

Note: If you are using both the IBM MQ Console, and token authentication with the REST API, the expiry interval is shared.

Attention: z/OS V 9.0.4

Before issuing 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 do this, 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.sh`. For example:

```
export WLP_USER_DIR=/var/mqm/web/installation1
```

For more information, see Create the Liberty server definition.

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. 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. V 9.0.4 From Version 9.0.4, you can configure when the tokens expire by using the **setmqweb** command. In IBM MQ Version 9.0.3, and earlier, you can configure when the tokens expire by editing the `mqwebuser.xml` file.

Procedure

V 9.0.4 Use one of the following methods to configure token expiry:

- From Version 9.0.4, use the **setmqweb properties** 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. The default value is 120 minutes.
- For Version 9.0.3 and earlier, edit the `mqwebuser.xml` file:
  1. Open the `mqwebuser.xml` file.

     The `mqwebuser.xml` file can be found in one of the following directories:
     - ULW On UNIX, Linux, and Windows: `MQ_DATA_DIRECTORY/web/installations/installationName/servers/mqweb`
     - z/OS On z/OS: `WLP_user_directory/servers/mqweb`

       where *WLP_user_directory* is the directory that was specified when the **crtmqweb.sh** script ran to create the mqweb server definition.
  2. Configure the LTPA token expiry interval by adding or editing the following line in the `mqwebuser.xml` file, within the <server> tags:

     ```
     <variable name="ltpaExpiration" value="time" />
     ```

     where *time* specifies the time, in minutes, before the LTPA token expires and the user is logged out. The default value is 120 minutes.

Configuring the administrative REST API gateway

V 9.0.4

By default, the administrative REST API gateway is enabled. 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 properties** command.

About this task

You must be a privileged user to complete this procedure.

You can view the current configuration of the administrative REST API gateway by using the following command:

```
dspmqweb properties -a
```

The `mqRestGatewayEnabled` field shows whether the gateway is enabled, and `mqRestGatewayQmgr` field shows the name of the default gateway queue manager. For more information, see dspmqweb.

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.

Attention: z/OS V 9.0.4

Before issuing 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 do this, 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.sh`. For example:

```
export WLP_USER_DIR=/var/mqm/web/installation1
```

For more information, see Create the Liberty server definition.

Procedure

- 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

V 9.0.4

By default, the mqweb server which hosts the IBM MQ Console and REST API has the messaging REST API enabled. You can configure whether messaging is enabled or disabled by using the **setmqweb properties** command.

Before you begin

You must be a privileged user to complete this procedure.

You can 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 or disabled. For more information, see dspmqweb.

To use the messaging REST API the caller must be authenticated to the mqweb server and must be a member of the `MQWebUser` role. The `MQWebAdmin` and `MQWebAdminRO` roles are not applicable for the messaging REST API. The caller must also be authorized through OAM/RACF. For more information about security for the REST API, see IBM MQ Console and REST API security.

Attention: z/OS V 9.0.4

Before issuing 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 do this, 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.sh`. For example:

```
export WLP_USER_DIR=/var/mqm/web/installation1
```

For more information, see Create the Liberty server definition.

Procedure

V 9.0.4 Use the following method to configure the messaging REST API:

- Use the **`setmqweb properties`** command:
  - Configure whether the messaging REST API is enabled, by using the following command:

    ```
    setmqweb properties -k mqRestMessagingEnabled -v enabled
    ```

    where *enabled* is the value `true` if you want the messaging REST API enabled, or `false` otherwise.

Tuning the mqweb server JVM

V 9.0.2

By default, the mqweb server Java virtual machine (JVM) uses platform-specific defaults for the minimum and maximum size of the heap. You might need to change the default values. For example, if a `java.lang.OutOfMemoryError` is thrown by the mqweb server, you must increase the maximum size of the heap. 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:

   - ULW On UNIX, Linux, and Windows: *MQ_DATA_DIRECTORY*/web/installations/*installationName*/servers/mqweb
   - z/OS On z/OS: *WLP_user_directory*/servers/mqweb

     where *WLP_user_directory* is the directory that was specified when the **`crtmqweb.sh`** script ran to create the mqweb server definition.
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. For example, the following line sets the minimum heap size to 512MB:

   ```
   -Xms512m
   ```
4. Restart the mqweb server by entering the following commands on the command line:

   ```
   endmqweb
   strmqweb
   ```

File structure of the IBM MQ Console and REST API installation component

V 9.0.1

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:

- ULW On UNIX, Linux, and Windows: `*MQ_DATA_DIRECTORY*/web/installations/*installationName*/`
- z/OS On z/OS: `*WLP_user_directory*`

  where *WLP_user_directory* is the directory that was specified when the **`crtmqweb.sh`** script ran to create the mqweb server definition.

Under this top-level directory, the following directories and files are present:

Directories and files	Description
`angular.persistence/`	Directory where the IBM MQ Console dashboard configuration is stored.
`servers/`	WebSphere Liberty Profile servers directory.
`servers/mqweb`	Directory that contains the mqweb server directory structure.
`servers/mqweb/logs`	Directory that contains logs for the mqweb server.
`servers/mqweb/logs/console.log`	Log of basic server status and operation messages.
`servers/mqweb/logs/ffdc`	First Failure Data Capture (FFDC) output directory.
`servers/mqweb/logs/messages.log`	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 `messages_*timestamp*.log`.
`servers/mqweb/logs/trace.log`	Log of trace from the mqweb server, including the IBM MQ Console and REST API. Older trace is stored in files that are called `trace_*timestamp*.log`. These files exist only if trace is enabled.
`servers/mqweb/logs/state`	Server-specific state.
`servers/mqweb/server.xml`	Main server configuration file. This file is read only. Edit the `mqwebuser.xml` file to override the default configuration.
`servers/mqweb/mqwebuser.xml`	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.

Directories and files	Description
`servers/mqweb/resources`	Directory that contains various server resources such as keystores.
`servers/mqweb/workarea`	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 user editable files depends on the operating system:

- ULW On UNIX, Linux, and Windows: `*MQ_INSTALLATION_PATH*/web`
- z/OS On z/OS: `*installation_directory*/web/`
  where *installation_directory* is the IBM MQ UNIX System Services Components installation path.

The following directory structure and files are present in this location:

Directories and files	Description
`bin/`	Directory that contains Liberty commands. You must be a privileged user 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.

Configuring IBM MQ client for HP Integrity NonStop Server

NSS Client

Use this information to help you to configure your HP Integrity NonStop Server installation to work with the IBM MQ client for HP Integrity NonStop Server.

About this task

If you are performing HP Integrity NonStop Server operations under TMF/Gateway, see the subtopics for information about how to configure the TMF/Gateway. Included are an overview of the Gateway process, configuring the Gateway to run under Pathway, and configuring the client initialization file to enable your IBM MQ client for HP Integrity NonStop Server to reach the TMF Gateway.

This section also contains specific HP Integrity NonStop Server information about granting permissions to channels.

Procedure

For information on how to configure your HP Integrity NonStop Server installation, see the following subtopics:

- "Gateway process overview"
- "Configuring Gateway to run under Pathway" on page 1414
- "Configuring the client initialization file" on page 1415
- "Granting permissions to channels" on page 1416

Related tasks:

"Configuring a client using a configuration file" on page 876
You configure your clients by using attributes in a text file. These attributes can be overridden by environment variables or in other platform-specific ways.

"Using IBM MQ environment variables" on page 896
You can use commands to display the current settings or to reset the values of IBM MQ environment variables.

Gateway process overview

NSS Client

The HP NonStop Transaction Management Facility (TMF) provides services to enable a gateway process to register as a resource manager. The TMF/Gateway process used by IBM MQ client for HP Integrity NonStop Server runs under Pathway.

IBM MQ client for HP Integrity NonStop Server registers a single gateway process for each queue manager that is coordinated by TMF, therefore you must configure a separate TMF/Gateway for each queue manager that is to participate in TMF coordinated units of work. This registration is so that each queue manager is an independent resource manager, and for administrative purposes, registering each queue manager once with HP NonStop TMF results in an easy to understand mapping.

For multiple installations of IBM MQ client for HP Integrity NonStop Server, you must nominate a single gateway process from one of these installations for each queue manager to be coordinated by TMF.

The interface to the gateway process supports any client at the same version or earlier.

For more information about administering the gateway process, see Administering IBM MQ client for HP Integrity NonStop Server.

Configuring Gateway to run under Pathway

NSS Client

TMF/Gateway is the interface between the HP NonStop Transaction Management Facility (TMF) and IBM MQ that enables TMF to be the transaction coordinator for IBM MQ transactions.

The TMF/Gateway provided for the IBM MQ client for HP Integrity NonStop Server converts transactions from TMF coordination into eXtended Architecture (XA) transaction coordination to communicate with the remote queue manager.

You must have one TMF/Gateway per queue manager that requires coordination, and client configuration is required so that the client can connect to the correct Gateway.

The TMF/Gateway can use all the mechanisms available to the client to communicate with a queue manager. Configure the TMF/Gateway in the way you would for your other applications.

The TMF/Gateway is not a HP Integrity NonStop Server process pair and is designed to run in a Pathway environment. The TMF/Gateway creates permanent resources within TMF, which it reuses on subsequent runs, therefore the TMF/Gateway must always be run under the same user authority.

Defining the serverclass

TMF/Gateway is hosted as a serverclass within a Pathway environment. To define the serverclass, you must set the following server attributes:

`PROCESSTYPE = OSS`
: Specifies the type of servers in the serverclass. The Gateway process is a multi-threaded OSS program. This attribute is mandatory, and must be set to `OSS`.

`MAXSERVERS = 1`
: Specifies the maximum number of server processes in this serverclass that can run at the same time. There can be only a single Gateway process for any queue manager. This attribute is mandatory and must be set to 1.

`NUMSTATIC = 1`
: Specifies the maximum number of static servers within this serverclass. The Gateway process must be run as a static server. This attribute is mandatory and must be set to `1`.

`TMF = ON`
: Specifies whether servers in this serverclass can lock and update data files that are audited by the TMF subsystem. The Gateway process participates in the TMF transactions of IBM MQ client applications therefore this attribute must be set to `ON`.

`PROGRAM =` *`OSS installation path`***`/opt/mqm/bin/runmqtmf`**
: For the IBM MQ client for HP Integrity NonStop Server, this attribute must be `runmqtmf`. This attribute must be the absolute OSS path name. Case is significant.

`ARGLIST = -m` ***`QMgr name`*** **`[,-c`** ***`channel name`*** **`][,-p`** ***`port`*** **`][,-h`** ***`host name`*** **`][,-n`** ***`max threads`*** **`]`**
: These attributes provide parameters to the Gateway process, where:

- *QMgrName* is the name of the queue manager for this Gateway process. If you are using a queue-sharing group (or other port distribution technology), this parameter must be targeted to a specific queue manager. This parameter is mandatory.
- *channel name* is the name of the server channel on the queue manager to be used by the Gateway process. This parameter is optional.
- *port* is the TCP/IP port for the queue manager. This parameter is optional.
- *host name* is the host name for the queue manager. This parameter is optional.

- *max threads* is the maximum number of worker threads that are created by the Gateway process. This parameter can be a value of 10 or greater. The lowest value that is used is 10 even if a value lower than 10 is specified. If no value is provided, the Gateway process creates up to a maximum of 50 threads.

Use the `-c`, `-p`, and `-h` attributes as an alternative method of providing connection information to the Gateway, in addition to that described in "Configuring the TMF/Gateway using environment variables." If you specify one or more, but not all of the `-c`, `-p`, and `-h` attributes, then those attributes that you do not specify default to the following values:

- *channel name* defaults to `SYSTEM.DEF.SVRCONN`
- *host name* defaults to `localhost`
- *port* defaults to `1414`

If any of the parameters you supply are invalid, the TMF/Gateway issues diagnostic message AMQ5379 to the error log and terminates.

OWNER = *ID*
The user ID under which the Gateway runs and that must be granted connect authority to the queue manager.

SECURITY = "*value*"
Specifies the users, in relation to the `Owner` attribute, who can access the Gateway from an IBM MQ client application.

`LINKDEPTH` and `MAXLINKS` must be configured with values appropriate for the expected number of IBM MQ client applications that might want to concurrently communicate with the Gateway. If these values are set too low, you might see occurrences of the error message AMQ5399 issued from client applications.

For more information about these server attributes, see the *HP NonStop TS/MP 2.5 System Management Manual*.

Configuring the TMF/Gateway using environment variables

One of the most commonly used methods to define the TMF/Gateway is to set the MQSERVER environment variable, for example:

```
ENV MQSERVER=channel name/transport/host name(listener port)
```

ENV at the beginning of the command is Pathway notation.

Configuring the client initialization file

NSS Client

If you are using the HP NonStop Transaction Management Facility (TMF), you must have an IBM MQ client initialization file to enable your IBM MQ client for HP Integrity NonStop Server to reach the TMF Gateway.

An IBM MQ client initialization file for HP Integrity NonStop Server can be held in a number of locations, for more information, see "Location of the client configuration file" on page 877.

For details of the contents of the configuration file, together with an example, see "Configuring a client using a configuration file" on page 876. Use the TMF stanza to specify the TMF queue manager and server details, for more information, see "TMF and TmfGateway stanzas" on page 895.

Here is an example of the entries for an IBM MQ client for HP Integrity NonStop Server:

```
TMF:
PathMon=$PSD1P

TmfGateway:
QManager=MQ5B
Server=MQ-MQ5B

TmfGateway:
QManager=MQ5C
Server=MQ-MQ5C
```

For more information about configuring a client using environment variables, see “Using IBM MQ environment variables” on page 896.

Granting permissions to channels

NSS Client

Granting permissions to channels on an IBM MQ client for HP Integrity NonStop Server is identical to other operating systems, however you must know the identification of the owner that the gateway is running under.

You can then use the identification of the owner of the gateway to grant appropriate permissions. The important difference is that granting permissions to queue manager channels is not under the authority of any application.

Use the setmqaut command to both to grant an authorization, that is, give an IBM MQ principal or user group permission to perform an operation, and to revoke an authorization, that is, remove the permission to perform an operation.

Configuring IBM MQ using Docker

Use this information to configure IBM MQ using Docker.

About this task

Docker allows you to package an IBM MQ queue manager or IBM MQ client application, with all of its dependencies, into a standardized unit for software development.

Changes to your application can be deployed to test and staging systems quickly and easily. This feature can be a major benefit to continuous delivery in your enterprise.

Procedure

For information on how to configure IBM MQ using Docker, see the following subtopics:

- Linux “Docker support on Linux systems” on page 1417
- “Planning your own IBM MQ queue manager image using Docker” on page 1417
- “Building a sample IBM MQ queue manager image using Docker” on page 1418
- “Running local binding applications in separate containers” on page 1421

Docker support on Linux systems

Linux

Information to consider if you are using Docker on a Linux system.

- The base image used by the Docker image must use a Linux operating system that is supported.
- You must use the IBM MQ installers to install the product inside the Docker image.
- For a list of supported packages, see IBM MQ rpm components for Linux systems.
- V 9.0.4 The following packages are not supported:
  - `MQSeriesBCBridge`
  - `MQSeriesRDQM`
- The queue manager data directory (`/var/mqm` by default) must be stored on a Docker volume which keeps persistent state.

  Important: You cannot use the union file system.

  You must either mount a host directory as a data volume, or use a data volume container. For more information, see Manage data in containers.
- You must be able to run IBM MQ control commands, such as **`endmqm`**, within the container.
- You must be able to get files and directories from within the container for diagnostic purposes.
- V 9.0.3 You can use namespacing to share the namespaces of the container for the queue manager with other containers, in order to locally bind applications to a queue manager running in separate containers. For more information, see "Running local binding applications in separate containers" on page 1421.

Planning your own IBM MQ queue manager image using Docker

Use this information to configure IBM MQ using Docker. There are several requirements to consider when running an IBM MQ queue manager in Docker. The sample Docker image provides a way to handle these requirements, but if you want to use your own image, you need to consider how these requirements are handled.

Process supervision

When you run a Docker container, you are essentially running a single process (PID 1 inside the container), which can later spawn child processes.

If the main process ends, Docker stops the container. An IBM MQ queue manager requires multiple processes to be running in the background.

For this reason, you need to make sure that your main process stays active as long as the queue manager is running. It is good practice to check that the queue manager is active from this process, for example, by performing administrative queries.

Populating `/var/mqm`

Docker containers must be configured with `/var/mqm` as a Docker volume.

When you do this, the directory of the volume is empty when the container first starts. This directory is usually populated at installation time, but installation and runtime are separate environments when using Docker.

V 9.0.3 To solve this, when your container starts, you can use the **`crtmqdir`** command to populate `/var/mqm` when it runs for the first time.

Building a sample IBM MQ queue manager image using Docker

Use this information to build a sample Docker image for running an IBM MQ queue manager in a Docker container.

About this task

Firstly, you build a base image containing an Ubuntu Linux file system and a clean installation of IBM MQ.

Secondly, you build another Docker image layer on top of the base, which adds some IBM MQ configuration to allow basic user ID and password security.

Finally, you run a Docker container using this image as its file system, with the contents of `/var/mqm` provided by a container-specific Docker volume on the host file system of Docker.

Procedure

For information on how to build a sample Docker image for running an IBM MQ queue manager in a Docker container, see the following subtopics:
- "Building a sample base IBM MQ queue manager image"
- "Building a sample configured IBM MQ queue manager image" on page 1419

Building a sample base IBM MQ queue manager image

In order to use IBM MQ in Docker, you need initially to build a base image with a clean IBM MQ installation. The following steps show you how to build a sample base image, using code hosted on GitHub.

Procedure

1. Install the prerequisite packages.

   These instructions make use of some Linux packages that you must install.
   - On Ubuntu:
     ```
     sudo apt-get install python git
     ```
   - On Red Hat Enterprise Linux:
     ```
     sudo yum install python git
     ```
2. Create a `downloads` directory by issuing the command `mkdir downloads`.
3. Download the IBM MQ server for Linux image, using Passport Advantage. See Installation using Electronic Software Download for more details.

   For example, select the `WS_MQ_V9.0.0_LINUX_ON_X86_64_IM.tar.gz` file, and place the file in the `downloads` directory that you have created.
4. Make the IBM MQ server for Linux image (`tar.gz`) file available on an HTTP or FTP server.

   The reason for this is to save space in the Docker image layers. Every instruction in a Docker file causes a new image layer to be created.

   If you use the **ADD** or **COPY** instructions, followed by a **RUN** instruction to install, then the files added or copied will be committed to a new image layer.

   Even if you delete the file in subsequent layers, the file still exists in the previous layer. For this reason, it is good practice to download and install within a single **RUN** command, which means the files need to be available on the network.

   For example, you can use Python to run an HTTP server, serving all files in your current directory:

```
pushd downloads
nohup python -m SimpleHTTPServer 8000 &
popd
```

5. Extract the sample files, for building a supported Docker image, from GitHub:

- V 9.0.0 For IBM MQ Version 9.0.0, issue the following command:

```
git clone -b mq-9-lts https://github.com/ibm-messaging/mq-docker mq-docker
```

- V 9.0.1 For IBM MQ Version 9.0.1, issue the following command:

```
git clone https://github.com/ibm-messaging/mq-docker mq-docker
```

6. Identify your local IP address.

   Your address is specific to your local environment, but should be available if you run the following command:

   ```
   ip addr show
   ```

   Note that `localhost` does not work.
7. Build the base IBM MQ image by issuing the following command, replacing the IP address and file name in the MQ_URL for the values that you have just identified: For example:

```
sudo docker build --tag mq --build-arg MQ_URL=http://10.0.2.15:8000/WS_MQ_V9.0.0.0_LINUX_ON_X86_64_IM.tar.gz mq-docker
```

Results

You now have a base Docker image with IBM MQ installed.

Building a sample configured IBM MQ queue manager image

Once you have built your generic base IBM MQ Docker image, you need to apply your own configuration to allow secure access. To do this, create your own Docker image, using the generic image as a parent. The following steps show you how to build a sample image, with a minimal security configuration.

Procedure

1. Create a new directory, and add a file called `config.mqsc`, with the following contents:

```
DEFINE CHANNEL(PASSWORD.SVRCONN) CHLTYPE(SVRCONN)
SET CHLAUTH(PASSWORD.SVRCONN) TYPE(BLOCKUSER) USERLIST('nobody') +
DESCR('Allow privileged users on this channel')
SET CHLAUTH('*') TYPE(ADDRESSMAP) ADDRESS('*') USERSRC(NOACCESS) DESCR('BackStop rule')
SET CHLAUTH(PASSWORD.SVRCONN) TYPE(ADDRESSMAP) ADDRESS('*') USERSRC(CHANNEL) CHCKCLNT(REQUIRED)
ALTER AUTHINFO(SYSTEM.DEFAULT.AUTHINFO.IDPWOS) AUTHTYPE(IDPWOS) ADOPTCTX(YES)
REFRESH SECURITY TYPE(CONNAUTH)
```

   Note that the preceding example uses simple user ID and password authentication. However, you can apply any security configuration that your enterprise requires.
2. Create a file called `Dockerfile`, with the following contents:

```
FROM mq
RUN useradd johndoe -G mqm && \
    echo johndoe:passw0rd | chpasswd
COPY config.mqsc /etc/mqm/
```

   where:
   - `johndoe` is the user ID that you want to add
   - `passw0rd` is the original password
3. Build your custom Docker image using the following command:

```
sudo docker build -t mymq .
```

   where "`.`" is the directory containing the two files you have just created.

   Docker then creates a temporary container using that image, and runs the remaining commands.

   The **`RUN`** command adds a user named `johndoe` with password `passw0rd` and the **`COPY`** command adds the `config.mqsc` file into a specific location known by the parent image.
4. Run your new customized image to create a new container, with the disk image you have just created.

   Your new image layer did not specify any particular command to run, so that has been inherited from the parent image. The entry point of the parent (the code is available on GitHub):

- Creates a queue manager
- Starts the queue manager
- Creates a default listener
- Then runs any MQSC commands from `/etc/mqm/config.mqsc.`

Issue the following commands to run your new customized image:

```
sudo docker run \
  --env LICENSE=accept \
  --env MQ_QMGR_NAME=QM1 \
  --volume /var/example:/var/mqm \
  --publish 1414:1414 \
  --detach \
  mymq
```

where the:

First `env` parameter
: Passes an environment variable into the container, which acknowledges your acceptance of the license for IBM IBM WebSphere MQ. You can also set the LICENSE variable to view to view the license.

  See IBM MQ license information for further details on IBM MQ licenses.

Second `env` parameter
: Sets the queue manager name that you are using.

Volume parameter
: Tells the container that whatever MQ writes to `/var/mqm` should actually be written to `/var/example` on the host.

  This option means that you can easily delete the container later, and still keep any persistent data. This option also makes it easier to view log files.

Publish parameter
: Maps ports on the host system to ports in the container. The container runs by default with its own internal IP address, which means that you need to specifically map any ports that you want to expose.

  In this example, that means mapping port 1414 on the host to port 1414 in the container.

Detach parameter
: Runs the container in the background.

Results

You have built a configured Docker image and can view running containers using the docker **`ps`** command. You can view the IBM MQ processes running in your container using the docker **`top`** command.

Attention: If your container is not shown when you use the docker **`ps`** command the container might have failed. You can see failed containers using the command docker **`ps -a`**.

The container ID will be shown by using the docker **`ps -a`** command, and was also printed when you issued the docker **`run`** command.

You can view the logs of a container using the docker **`logs ${CONTAINER_ID}`** command.

A common problem is that **`mqconfig`** indicates that certain kernel settings on the Docker host are not correct. Kernel settings are shared between the Docker host and containers, and need to be set correctly (see Hardware and software requirements on UNIX and Linux systems.

For example, the maximum number of open files can be set using the command **sysctl fs.file-max=524288**.

Running local binding applications in separate containers

With the addition of process namespace sharing between containers in Docker. You can now run applications that require a local binding connection to IBM MQ in separate containers from the IBM MQ queue manager. This functionality is supported in IBM MQ Version 9.0.3 and later queue managers.

About this task

You must adhere to the following restrictions:

- Docker version 1.12 or later must be used.
- You must share the containers PID namespace using the `--pid` argument.
- You must share the containers IPC namespace using the `--ipc` argument.
- You must either:
  1. Share the containers UTS namespace with the host using the `--uts` argument, or
  2. Ensure the containers have the same hostname using the `-h` or `--hostname` argument.
- You must mount the IBM MQ data directory in a volume that is available to the all containers under the `/var/mqm` directory.

You can try this functionality out, by completing the following steps on a Linux system that already has Docker 1.12 or later installed.

The following example uses the sample IBM MQ Docker container image. You can find details of this image on Github.

Procedure

1. Create a temporary directory to act as your volume, by issuing the following command:

```
mkdir /tmp/dockerVolume
```

2. Create a queue manager (`QM1`) in a container, with the name `sharedNamespace`, by issuing the following command:

```
docker run -d -e LICENSE=accept -e MQ_QMGR_NAME=QM1 --volume /tmp/dockerVol:/mnt/mqm
--uts host --name sharedNamespace ibmcom/mq
```

3. Create a second container called `secondaryContainer`, which shares the namespace and volume of the first container, and has a queue manager (`QM2`) running on it, by issuing the following command:

```
docker run -d -e LICENSE=accept -e MQ_QMGR_NAME=QM2 --volumes-from sharedNamespace
--pid container:sharedNamespace --ipc container:sharedNamespace --uts host
--name secondaryContainer ibmcom/mq
```

4. Run the **dspmq** command on the second container, to see the status for both queue managers, by issuing the following command:

```
docker exec secondaryContainer dspmq
```

5. Run the following command to process MQSC commands against the queue manager running on the other container:

```
docker exec -it secondaryContainer runmqsc QM1
```

Results

You now have local applications running in separate containers.

If you do not see the result you expected, see “Troubleshooting your namespace applications” on page 1422 for more information.

Troubleshooting your namespace applications

When using shared namespacing, you must ensure that you share all namespaces (IPC, PID and UTS/hostname) and mounted volumes, otherwise your applications will not work.

See "Running local binding applications in separate containers" on page 1421 for a list of restrictions you must follow.

If your application does not meet all the restrictions listed, you could encounter problems where the container starts, but the functionality you expect does not work.

The following list outlines some common causes, and the behavior you are likely see if you have forgotten to meet one of the restrictions.

- If you forget to share a namespace (UTS/PID/IPC) or set the hostname of the containers as the same, but mount the volume, your container will be able to see the queue manager, but not interact with the queue manager.
  - For **dspmq** commands, you see the following:

```
docker exec container dspmq

QMNAME(QM1)                    STATUS(Status not available)
```

  - For **runmqsc** commands, or other commands that try to connect to the queue manager, you are likely to receive an AMQ8146 error message:

```
docker exec -it container runmqsc QM1
5724-H72 (C) Copyright IBM Corp. 1994, 2017.
Starting MQSC for queue manager QM1.
AMQ8146: IBM MQ queue manager not available
```

- If you share all the required namespace, but you do not mount a shared volume to the `/var/mqm` directory, and have a valid IBM MQ data path then your commands also receive AMQ8146 error messages.

  However, **dspmq** is not able to see your queue manager at all, and instead returns a blank response:

```
docker exec container dspmq
```

- If you share all the required namespace but you do not mount a shared volume to the `/var/mqm` directory, and you do not have a valid IBM MQ data path (or no IBM MQ data path) you see various errors as the data path is a key component of an IBM MQ installation. Without the data path, IBM MQ cannot operate.

  If you run any of the following commands, and see responses similar to those shown below, you should verify that you have mounted the directory or created an IBM MQ data directory:

```
docker exec container dspmq
'No such file or directory' from /var/mqm/mqs.ini
AMQ6090: IBM MQ was unable to display an error message FFFFFFFF.
AMQffff

docker exec container dspmqver
AMQ7047: An unexpected error was encountered by a command. Reason code is 0.

docker exec container mqrc
/build/slot1/p900_P/src/cmd/tools/mqrc/mqrc.c[1152]
lpiObtainQMDetails --> 545261715

docker exec container crtmqm QM1
AMQ8101: IBM MQ error (893) has occurred.

docker exec container strmqm QM1
AMQ6239: Permission denied attempting to access filesystem location '/var/mqm'.
AMQ7002: An error occurred manipulating a file.

docker exec container endmqm QM1
AMQ8101: IBM MQ error (893) has occurred.

docker exec container dltmqm QM1
AMQ7002: An error occurred manipulating a file.
```

```
docker exec container strmqweb
/build/slot1/p900_P/src/cmd/tools/mqrc/mqrc.c[1152]
lpiObtainQMDetails --> 545261715
```

Required configuration for the MFT REST API

V 9.0.5

Steps you need to take to configure Managed File Transfer to use the REST API

In the `mqwebuser.xml` file, set the:
- **`mqRestMftEnabled`** property must to *true*

  Note: You must restart the `mqweb` server if you change the value of this property.
- **`mqRestMftCoordinationQmgr`** property to an appropriate MFT coordination queue manager running locally on the machine where the `mqweb` server is running.

  For a queue manager to act as an MFT coordination queue manager for the REST API, you must run the:
  - **`fteSetupCoordination`** command on the same local queue manager that was set for the **`mqRestMftCoordinationQmgr`** property in the `mqwebuser.xml` file.

    This command generates an MQSC file that contains definitions of IBM MQ objects. Note that the command can be run on any machine that has MFT installed.
  - MQSC file generated in the preceding step, should be run against the coordination queue manager, to create the required IBM MQ objects.

See MFT REST API configuration parameters for more information.

Related information:

Configuring MFT REST API security

Administering the MFT REST API

GET - list of transfers

GET - transfer status

Configuring IBM MQ for use with IBM Cloud Product Insights service in IBM Cloud

Linux Windows V 9.0.2

Connect your IBM MQ queue manager to the IBM Cloud Product Insights service in IBM Cloud (formerly Bluemix®) to report and view queue manager startup and usage information.

Before you begin

Before you configure your IBM MQ queue managers to use a IBM Cloud service, you must have an IBM Cloud (formerly Bluemix) account. To create your account, see Sign up for IBM Cloud.

About this task

By using IBM Cloud Product Insights, you can connect your on-premises IBM products to your Product Insights service instance in IBM Cloud and see all the registered products in your organization in a single dashboard.

Linux Windows V 9.0.2 From IBM MQ Version 9.0.2, you can configure and connect your Linux and Windows queue managers to your Product Insights service instance and see their startup and usage information.

Note:

z/OS V 9.0.3 To connect to IBM Cloud Product Insights, you need an IBM MQ Advanced for z/OS VUE queue manager Version 9.0.3 or later. For more information, see "Configuring IBM MQ Advanced for z/OS VUE for use with IBM Cloud Product Insights service in IBM Cloud (formerly Bluemix)" on page 1592.

AIX V 9.0.4 From IBM MQ Version 9.0.4, you can configure your IBM MQ for AIX queue manager to connect to IBM Cloud Product Insights. Follow the instructions in the next two topics to configure and connect your AIX queue manager.

V 9.0.4 Use the following attributes with the V 9.0.5 `ReportingService` (formerly `BluemixRegistration`) stanza in the `qm.ini` file:

`APIKeyFile`
Location of the text file with the Product Insights service instance **`APIKey`** value.

`ServiceURL`
The IBM Cloud 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 internet.

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 Product Insights service, the following information is reported to IBM Cloud:

- 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
- Usage information for the IBM MQ queue manager

You can monitor your queue manager usage metrics in your Product Insights service instance dashboard.

For more information, see IBM Cloud Product Insights service in IBM Cloud and the IBM Cloud Product Insights developer center.

Procedure

1. Create an IBM Cloud Product Insights service instance on IBM Cloud (formerly Bluemix)
2. Configure a queue manager for use with the IBM Cloud Product Insights service instance on IBM Cloud (formerly Bluemix)

Related information:

Video: IBM MQ to Product Insights Service in IBM Cloud (YouTube)

Creating an IBM Cloud Product Insights service instance on IBM Cloud (formerly Bluemix)

Linux Windows V 9.0.2

Use your IBM Cloud account to create a Product Insights service instance and copy the security credentials that you need to register your queue managers to the service.

About this task

The Product Insights service instance that you are creating is associated to a single organization and space within IBM Cloud and has unique credentials. You must create at least one organization and space but if you need to separate the data you can create multiple spaces within an organization and have a service in each space.

z/OS See "Creating an IBM Cloud Product Insights service instance on IBM Cloud (formerly Bluemix)" on page 1594, for information on carrying out this process on z/OS.

Procedure

1. Create an instance of the IBM Cloud Product Insights service by following these steps:
   a. Log in to IBM Cloud.
   b. In the menu bar of the IBM Cloud user interface, click **Catalog**.
   c. Click **Integrate** in the **Platform** section, then click the **Product Insights** tile. The Product Insights service page is displayed.
   d. Optional: Specify a service name. The **Service** name field is available when you are logged in.
   e. Optional: Specify a name for the credentials that you want to use for the connection with IBM MQ. The **Credentials** name field is available when you are logged in.
   f. Click **Create**. Your IBM Cloud Product Insights service instance is created and the **Getting Started** tab of your Product Insights service dashboard is shown.
2. Copy the security credentials for connecting your queue manager to the IBM Cloud Product Insights service by following these steps:
   a. Click the **Service credentials** tab. The list of credentials for your on-premises products is displayed and includes the service credentials that you specified for your new Product Insights service.
   b. Click **View credentials** to show the API host and the API key values that you need to use to connect your IBM MQ queue manager to the Product Insights service instance.
   c. Copy the API host from the **Service credentials** tab so that you can add it to the `qm.ini` file for your queue manager. You need this in step 8 of the next task.
   d. Copy the API key from the **Service credentials** tab so that you can add it to your API key file when you configure your IBM MQ queue manager in step 7 of the next task.

What to do next

Follow the instructions for "Configuring a queue manager for use with the IBM Cloud Product Insights service instance on IBM Cloud (formerly Bluemix)" on page 1426.

Configuring a queue manager for use with the IBM Cloud Product Insights service instance on IBM Cloud (formerly Bluemix)

Linux Windows V 9.0.2

Set up security and IBM Cloud registration information for your queue manager, and then connect to the Product Insights service instance that you already created.

Before you begin

You created an IBM Cloud Product Insights service instance as described in "Creating an IBM Cloud Product Insights service instance on IBM Cloud (formerly Bluemix)" on page 1425.

z/OS For z/OS, see "Configuring a z/OS queue manager for use with the IBM Cloud Product Insights service instance on IBM Cloud (formerly Bluemix)" on page 1595.

About this task

Your IBM Cloud Product Insights service instance dashboard shows data for only the queue managers that are configured to include the security and IBM Cloud registration information.

Procedure

1. Download the `DigiCert Global Root CA` and `DigiCert SHA2 Secure Server CA` certificates and save them in a local folder. When your queue manager connects to the Product Insights service in IBM Cloud, the IBM Cloud public certificates are required by the queue manager to verify the authenticity of the service. Download the certificates in one of the following ways:
   a. Use CURL to download the certificates from command line:
   ```
   curl -o DigiCertGlobalRootCA.crt https://www.digicert.com/CACerts/DigiCertGlobalRootCA.crt
   ```
   and
   ```
   curl -o DigiCertSHA2SecureServerCA.crt https://www.digicert.com/CACerts/DigiCertSHA2SecureServerCA.crt
   ```
   b. Use your browser to download the certificates from the DigiCert website.
      - Find the row in the table with the certificate name `DigiCert Global Root CA`, right click **Download**, then **Save link as** or **Save target as**, depending on your browser. Select the location where you want to save the certificate.
      - Find the row in the table with the certificate name `DigiCert SHA2 Secure Server CA`, right click **Download**, then **Save link as** or **Save target as**, depending on your browser. Select the location where you want to save the certificate.

   Make a note of the location where you downloaded the certificates to. You can add the downloaded certificates to the keystore for your queue manager, in step 7.
2. 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 UNIX systems). The file must contain only the **`API key`** itself, not a JSON payload, for example `d9c11b45-4dda-4de4-c0b2-2e4e1004dc64`.
3. Create the queue manager, for example, *QM1*. For more information, see Creating and managing queue managers on Multiplatforms.
4. Start the queue manager *QM1*. For more information, see Starting a queue manager.
5. Remember to set up your IBM MQ command line environment before you run IBM MQ commands. Run the **`setmqenv`** command.

   AIX V 9.0.4 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
```

6. Create an SSL truststore for the queue manager *QM1*.

   AIX V 9.0.4 Begin creating the truststore, on AIX:Amend for AIX

   ```
   runmqckm -keydb -create -db MQ data directory/qmgrs/QM1/ssl/key.kdb -pw password -type cms -expire 30 -stash
   ```

   Linux On Linux:

   ```
   runmqckm -keydb -create -db MQ data directory/qmgrs/QM1/ssl/key.kdb -pw password -type cms -expire 30 -stash
   ```

   Windows On Windows:

   ```
   runmqckm -keydb -create -db "MQ data directory\qmgrs\QM1\ssl\key.kdb" -pw password -type cms -expire 30 -stash
   ```

7. Add the digital certificates that you downloaded in step 1 on page 1426, to the queue manager's truststore.

   AIX V 9.0.4 On AIX:

   ```
   runmqckm -cert -add -db MQ data directory/qmgrs/QM1/ssl/key.kdb -pw password -type cms -label DigiCertGlobalRootCA
   -file Download_location/DigiCertGlobalRootCA.crt -format ascii -trust enable

   runmqckm -cert -add -db MQ data directory/qmgrs/QM1/ssl/key.kdb -pw password -type cms -label DigiCertSHA2SecureServ
   -file Download_location/DigiCertSHA2SecureServerCA.crt -format ascii -trust enable
   ```

   Linux On Linux:

   ```
   runmqckm -cert -add -db MQ data directory/qmgrs/QM1/ssl/key.kdb -pw password -type cms -label DigiCertGlobalRootCA
   -file Download_location/DigiCertGlobalRootCA.crt -format ascii -trust enable

   runmqckm -cert -add -db MQ data directory/qmgrs/QM1/ssl/key.kdb -pw password -type cms -label DigiCertSHA2SecureServ
   -file Download_location/DigiCertSHA2SecureServerCA.crt -format ascii -trust enable
   ```

   Windows On Windows:

   ```
   runmqckm -cert -add -db "MQ data directory\qmgrs\QM1\ssl\key.kdb" -pw password -type cms -label DigiCertGlobalRootCA
   -file "Download_location\DigiCertGlobalRootCA.crt" -format ascii -trust enable

   runmqckm -cert -add -db "C:\ProgramData\IBM\MQ\qmgrs\QM1\ssl\key.kdb" -pw password -type cms -label DigiCertSHA2Secu
   -file "Download_location\DigiCertSHA2SecureServerCA.crt" -format ascii -trust enable
   ```

8. Add the new `ReportingService` stanza with the `apikeyfile` path to the queue manager's `qm.ini` file:

   V 9.0.5

   ```
   ReportingService:
      APIKeyFile=APIKey file location/apikeyfile.txt
   ```

9. Add the **`API host`** value to the `qm.ini` file. The `ReportingService` stanza section now contains path to the `apikeyfile` and the **`API host`** (**`ServiceURL`**) values:

   V 9.0.5

   ```
   ReportingService:
      APIKeyFile=APIKey file location/apikeyfile.txt
      ServiceURL=https://productinsights-api.ng.bluemix.net
   ```

   Save and exit the `qm.ini` file.

10. 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 Product Insights service.

11. View the information about the queue manager *QM1* in your Product Insights 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 Product Insights service. The usage information is updated every 15 minutes.
12. Optional: Stop a queue manager from reporting to the Product Insights service, by removing the `ReportingService` stanza from the queue manager's `qm.ini` file and restart the queue manager.
13. 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 Product Insights service. Amend for AIX

    AIX V 9.0.4 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 Product Insights service instance and configured your queue manager to connect to the instance. You can see the information about your queue manager in the Product Insights service instance dashboard.

Connecting to IBM Cloud Product Insights in IBM Cloud through an HTTP proxy

V 9.0.4

If your queue manager is running on a system that does not have direct access to the internet, you can use an HTTP proxy that your organization provides to connect to your IBM Cloud Product Insights service instance in IBM Cloud. Alternatively, you might use the gateway that is provided by the Product Insights service.

Before you begin

You have created your Product Insights service instance.

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 Product Insights service instance in IBM Cloud through an HTTP proxy that is provided by your organization.

You can also connect your queue manager to a Product Insights service instance through the gateway provided by Product Insights. See the Product Insights developer center information on how to use a Product Insights gateway..

Procedure

- Add a service proxy attribute to the IBM Cloud 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.

V 9.0.5

```
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.

Troubleshooting the connection to Product Insights

V 9.0.4

Troubleshooting advice for errors that you might encounter when you are connecting your queue manager to a IBM Cloud Product Insights service instance.

Queue manager cannot register with or upload usage metrics to the configured Product Insights 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 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.

Configuring IBM MQ for use with Salesforce push topics and platform events

Linux V 9.0.2

Use this information to set up security and connections to Salesforce and your IBM MQ network by configuring and then running the IBM MQ Bridge to Salesforce.

Before you begin

- IBM MQ Bridge to Salesforce is available on Linux Linux for System x (64 bit). The bridge is not supported for connecting to queue managers that are running on IBM WebSphere MQ Version 6.0 and earlier.
- Install the **`MQSeriesSFBridge`** package. For more information, see Installing IBM MQ server on Linux.

About this task

Salesforce is a cloud based, customer relationship management platform. If you are using Salesforce to manage customer data and interactions, at IBM MQ Version 9.0.2 you can use the IBM MQ Bridge to Salesforce to subscribe to Salesforce push topics and platform events that can then be published to your IBM MQ queue manager. Applications that connect to that queue manager can consume the push topic and platform event data, in a useful way.

V 9.0.4 From IBM MQ Version 9.0.4, you can also use the bridge to create event messages for platform events in Salesforce.

For an overview of the IBM MQ Bridge to Salesforce, see the diagram in Figure 1.

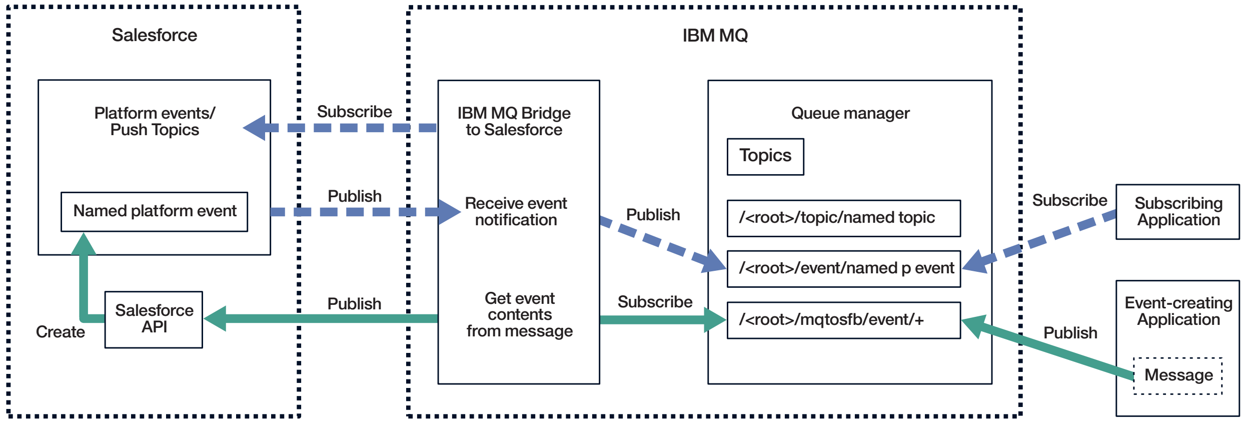

Salesforce
Platform events/ Push Topics
Named platform event
Salesforce API
Create
Subscribe
Publish
Publish
IBM MQ
IBM MQ Bridge to Salesforce
Receive event notification
Get event contents from message
Publish
Subscribe
Queue manager
Topics
/<root>/topic/named topic
/<root>/event/named p event
/<root>/mqtosfb/event/+
Subscribe
Subscribing Application
Publish
Event-creating Application
Message

Figure 174. IBM MQ Bridge to Salesforce

Push topics are queries that you define to use the `Force.com` Streaming API to receive notifications for changes to records in Salesforce. For more information on configuring push topics and how to use the Streaming API, see Introducing Streaming API and Working with PushTopics.

Platform events are customizable event messages that can be defined to determine the event data that the `Force.com` platform produces or consumes. For more information on platform events and the difference between Salesforce events, see Enterprise messaging platform events and What is the difference between the Salesforce events.

- To create the configuration for subscribing to push topics and platform events, see "Configuring the IBM MQ Bridge to Salesforce" on page 1431.
- V 9.0.4 To create the configuration for creating event messages for Salesforce platform events, see "Creating event messages for Salesforce platform events" on page 1435.

You can monitor the data from the bridge in two ways, through the IBM MQ Console and by using the **-p** parameter with the **amqsrua** command. One set of data is published for the overall bridge status:

- Total push topic messages that are processed in an interval (under the `STATUS/PUSHTOPIC` tree).
- Number of push topics that are seen in this interval.
- Total platform events that are processed in an interval (under the `STATUS/PLATFORM` tree).
- Number of platform events that are seen in this interval.
- V 9.0.4 Total number of IBM MQ created platform events that are processed in an interval (under the `STATUS/MQPE` tree).
- V 9.0.4 Unique number of IBM MQ created platform events that are seen in this interval.
- V 9.0.4 Failed number of publications of IBM MQ created platform events that are seen in this interval.

For each configured Salesforce topic, a further message is published. The IBM MQ topic uses the full Salesforce topic name and the `/event` or `/topic` in the object name:

- Number of messages that are processed in an interval.

To configure the IBM MQ Console to monitor the bridge data, see the steps 9 and 10 in the next task Configuring the IBM MQ Bridge to Salesforce. For information on using the **amqsrua** command, see Monitoring the IBM MQ Bridge to Salesforce.

Follow the steps in these tasks to configure and run the IBM MQ Bridge to Salesforce:

Procedure

1. Configure the IBM MQ Bridge to Salesforce.
2. V 9.0.4 Create event messages for Salesforce platform events.
3. Run the IBM MQ Bridge to Salesforce.

Related information:

runmqsfb (run IBM MQ Bridge to Salesforce)

Tracing the IBM MQ Bridge to Salesforce

Configuring the IBM MQ Bridge to Salesforce

Linux V 9.0.2

You can configure IBM MQ and enter IBM MQ Bridge to Salesforce parameters to create the configuration file and connect Salesforce push topics and platform events to your IBM MQ queue manager.

Before you begin

- You installed the **MQSeriesSFBridge** package in your IBM MQ installation on an x86-64 Linux platform.

About this task

This task takes you through the minimal setup that is needed to create the IBM MQ Bridge to Salesforce configuration file and successfully connect to Salesforce and IBM MQ so that you can subscribe to Salesforce push topics and platform events. For more information on the meaning and options for all the parameters, see the runmqsfb command. You must consider your own security requirements and customize the parameters appropriate to your deployment.

V 9.0.4 To create the configuration for creating event messages for Salesforce platform events, see "Creating event messages for Salesforce platform events" on page 1435.

Subscribing to Salesforce push topics and platform events

When the IBM MQ Bridge to Salesforce establishes connections to both Salesforce and IBM MQ, it creates subscriptions to Salesforce push topics and platform events. The push topic or platform event name that the bridge wants to subscribe to, must be included in the configuration file or added in the command line before the connection is made.

One of the configuration attributes is the root of the IBM MQ topic tree and the events are published beneath this root. The bridge accesses this root and adds the full Salesforce topic name, for example, `/MQ/SF/ROOT/topic/EscalatedCases`. The monitoring topic and applications that are connecting to IBM MQ might look for push topics under `/topic/EscalatedCases` and platform events under `/event/NewCustomer__e`.

The published message contains control information and the data structure that contains the requested data fields. For push topics, the data structure is an **sobject** and for platform events, the structure is **payload**. The bridge cannot subscribe to a topic or an event if they are not defined in Salesforce. If the bridge encounters an error when it tries to subscribe to a topic, the bridge stops.

A topic object does not need to be defined in IBM MQ but suitable authorities must exist, based on the closest parent element in the tree. The republished message contains only the relevant data structure from the original message by default. The control information is removed. For platform events, the publication has a payload structure. The **`Publish control data with the payload`** configuration option in the **Behaviour of bridge program** set of configuration parameters enables the republication of the entire message, including the control data. For more information, see Configuration parameters.

Each push topic and platform event has an associated *ReplayID* on publication from Salesforce. The *ReplayID* can be used to request the starting point for publication when the connection is made to the server. Salesforce maintains a history for up to 24 hrs and allows the bridge to not miss recent push topics and platform events even if it was not started at the time when they are generated. The bridge supports two quality of service modes:

At-most-once
: The bridge does not use the *ReplayId* for restart. After restart of the bridge, only newly generated push topics and platform events are processed. Applications must be prepared to deal with missing publications. The *ReplayId* is still tracked by the bridge and hardened to a queue, so the bridge can be restarted with the other quality of service and know the current state.

At-least-once
: The *ReplayId* is tracked by the bridge and hardened to a queue. On restart of the bridge, the persisted *ReplayId* is used to request the starting point for publications from the server. Provided the gap was no more than 24 hours, older publications are sent. The *ReplayId* for a topic is not hardened on every message. It is written in a persistent message at regular intervals and when the bridge is shut down. Applications must be prepared to see duplicate publications.

The *ReplayId* is written as a message to a newly defined queue. You must define this queue, **`SYSTEM.SALESFORCE.SYNCQ`**, before the bridge is started. If the **`SYSTEM.SALESFORCE.SYNCQ`** does not exist, the bridge does not continue, regardless of the quality of service mode. An MQSC script is provided for creating the queue with relevant attributes. The queue must be configured with the `DEFSOPT(EXCL) NOSHARE` option to ensure that only one instance of the bridge program can update the **`SYSTEM.SALESFORCE.SYNCQ`** queue.

V 9.0.4 To create the configuration for creating event messages for platform events, see "Creating event messages for Salesforce platform events" on page 1435.

Procedure

1. Create and start a queue manager.
   a. Create a queue manager, for example SQM1.
      ```
      crtmqm SQM1
      ```
   b. Start your queue manager.
      ```
      strmqm SQM1
      ```
2. Optional:

   Note: To use existing login and security Salesforce credentials and self-signed certificate, skip to step 4.
   Create a security token for your Salesforce account.
   a. Log in to your Salesforce account.
   b. Create or reset your security token by following the steps in the help article Salesforce help: Reset your security token.
3. Create a self-signed security certificate in Salesforce.
   a. Select **Security controls** from the **Administer** menu of your **Force.com Home** page, then **Certificate and Key Management**. The Certificate and Key Management page opens.
   b. Click **Create Self-Signed certificate**. The Certificates page opens.

c. Enter a name for the certificate in the **Label** field, press Tab, then click **Save**. The Certificate and Key Detail information is displayed.
d. Click **Back to list: Certificates and keys**.
e. Click **Export to Keystore**.
f. Enter a password for the keystore, then click **Export**.
g. Save the exported keystore to your local file system.

4. Use the IBM Key Management GUI to open the keystore you exported from Salesforce and populate the signer certificates.
   a. Run the **strmqikm** command to open the IBM Key Management GUI. For more information, see Using runmqckm, runmqakm, and strmqikm to manage digital certificates.
   b. Click **Open a key database file** and browse to the location of the Salesforce keystore.
   c. Click **Open**, make sure to select **JKS** from the **Key database type** options, then click **OK**.
   d. Enter the password that you created for the keystore in step 3f, then click **OK**.
   e. Select **Signer Certificates** from the **Key database content** options.
   f. Click **Populate**.
   g. Select the **Verisign Inc.** check box from the **Add CA Certificates** list, then click **OK**.
5. Optional: Generate OAuth consumer key and secret by creating an app connection for IBM MQ Bridge to Salesforce in your Salesforce account. You need the **Consumer Key** and **Consumer Secret** codes when you are using the IBM MQ Bridge to Salesforce in production environments.
   a. Select **Create**, then **Apps** from the **Build** menu of your **Force.com Home** page. The Apps page opens.
   b. Click **New** from the **Connected Apps** section. The New Connected App page opens.
   c. Enter a name for your IBM MQ Bridge to Salesforce in the **Connected App Name**, for example **MQBridgeToSalesforce**.
   d. Enter the **API Name**. If you tab through to the next field, the **Connected App Name** is copied into the **API Name** name field.
   e. Enter your **Contact Email**.
   f. Select the **Enable OAuth Settings** option in the **API (Enable OAuth Settings)** section. Further options in that section are then presented.
   g. Add your **Callback URL**, for example `https://www.ibm.com`.
   h. Select the **Full access (full)** option from the **Available OAuth Scopes** list in the **Selected OAuth Scopes** subsection, then click **Add**, to add full access to the **Selected OAuth Scopes** list.
   i. Click **Save**.
   j. Click **Continue**.
   k. Take note of your **Consumer Key** and **Consumer Secret** codes.
6. Create the required synchronization queue on the queue manager.
   ```
   cat /opt/mqm/mqsf/samp/mqsfbSyncQ.mqsc | runmqsc SQM1
   ```
   The synchronization queue maintains event state across application or queue manager restarts. The queue depth can be small as only a single message is expected on the queue. Only one instance of the bridge can run at a time against this queue, so the default options are set for exclusive access.
7. Create a configuration file with connection and security parameters for IBM MQ, Salesforce, and the IBM MQ Bridge to Salesforce behavior.
   ```
   runmqsfb -o new_config.cfg
   ```
   The existing values are shown inside the brackets. Press Enter to accept existing values, press Space then Enter to clear values, and, type, then Enter to add new values.
   a. Enter values for the connection to queue manager SQM1: Minimum values that are needed for the connection are queue manager name, IBM MQ base topic root, and channel name.

```
Connection to Queue Manager
---------------------------
Queue Manager or JNDI CF   : []SQM1
MQ Base Topic              : []/sf
MQ Channel                 : []A channel you have defined or for example SYSTEM.DEF.SVRCONN
MQ Conname                 : []
```

V 9.0.4 `MQ Publication Error Queue : [SYSTEM.SALESFORCE.ERRORQ]` MQ CCDT URL : [] JNDI implementation class : [com.sun.jndi.fscontext.RefFSContextFactory] JNDI provider URL : [] MQ Userid : [] MQ Password : []

Note: Channel name is not required if you are connecting locally. You don't have to provide the queue manager name and base topic in the configuration file as they can be included on the command line later, when you run the bridge.

b. Enter values for connection to Salesforce: Minimum values that are needed for the connection are Salesforce user ID, password, security token, and login endpoint. In production environments, you can add the consumer key and secret for OAuth security.

```
Connection to Salesforce
------------------------
Salesforce Userid (reqd)   : []salesforce_login_email
Salesforce Password (reqd) : []salesforce_login_password
Security Token (reqd)      : []Security_Token
Login Endpoint             : [https://login.salesforce.com]
Consumer ID                : []
Consumer Secret Key        : []
```

c. Enter values for certificate stores for TLS connections: Minimum values that are needed for TLS connections are the path to the keystore for TLS certificates and keystore password. If no trusted store path or password is provided, the keystore and password parameters are used for the trusted store and password. If you are using TLS for your IBM MQ queue manager connection, you can use the same keystore.

```
Certificate stores for TLS connections
--------------------------------------
Personal keystore for TLS certificates : []path_to_keystore, for example: /var/mqm/qmgrs/SQM1/ssl/key.jks
Keystore password          : []keystore_password
Trusted store for signer certificates : []
Trusted store password     : []
Use TLS for MQ connection  : [N]
```

d. Enter values to configure the behavior of the IBM MQ Bridge to Salesforce: You do not have to change or provide any of these values but if you know your push topic or platform event names, add them here. They can also be added later, in the command line, when you are ready to run the bridge. You must specify the log file, in the configuration file or on the command line.

```
Behaviour of bridge program
---------------------------
PushTopic Names          : []
Platform Event Names     : []
MQ Monitoring Frequency  : [30]
At-least-once delivery? (Y/N) : [Y]
```

V 9.0.4 `Subscribe to MQ publications for platform events? (Y/N) : [N]` Publish control data with the payload? (Y/N) : [N] Delay before starting to process events : [0] Runtime logfile for copy of stdout/stderr : []

8. Optional: Create the IBM MQ service to control the execution of the program. Edit the sample `mqsfbService.mqsc` file to point to the newly created configuration file and make any other changes to the command parameters.

   ```
   cat modified mqsfbService.mqsc | runmqsc SQM1
   ```

9. V 9.0.1 Optional: Follow instructions in Getting started with the IBM MQ Console to set up the IBM MQ Console.

10. Optional:

    Note: Before you can see any data about the bridge in MQ Console, you must run the bridge at least once so that when it is started, it makes the connections to Salesforce and IBM MQ. The meta-topics for the bridge are published at bridge start up.
    Add and configure widgets in your IBM MQ Console instance to view Salesforce data.
    a. Click **Add widget**. The new widget opens.
    b. Select **Charts**
    c. Click **Configure widget** icon in the title bar of the new widget.
    d. Optional: Enter a **Widget title**.
    e. Select **Salesforce Bridge**, from the **Resource to monitor**, **Source** drop-down menu.
    f. Click **Save**.

Results

You created the configuration file that the IBM MQ Bridge to Salesforce uses to subscribe to Salesforce push topics and platform events and publish them to your IBM MQ network.

What to do next

Work through the steps for "Running the IBM MQ Bridge to Salesforce" on page 1441.

Related information:

runmqsfb (run IBM MQ Bridge to Salesforce)

Tracing the IBM MQ Bridge to Salesforce

Monitoring the IBM MQ Bridge to Salesforce

Creating event messages for Salesforce platform events

Linux V 9.0.4

You can configure IBM MQ and enter `IBM MQ Bridge to Salesforce parameters` to create the configuration file and use the bridge to create event messages for Salesforce platform events.

Before you begin

- You installed the **`MQSeriesSFBridge`** package in your IBM MQ installation on an x86-64 Linux platform.

About this task

This task takes you through the minimal setup that is needed to create the IBM MQ Bridge to Salesforce configuration file and successfully connect to Salesforce and IBM MQ so that you can create event messages for Salesforce platform events. For more information on the meaning and options for all the parameters, see the runmqsfb command. You must consider your own security requirements and customize the parameters appropriate to your deployment.

To create the configuration for subscribing to push topics and platform events, see "Configuring the IBM MQ Bridge to Salesforce" on page 1431.

Creating event messages for Salesforce platform events

From IBM MQ Version 9.0.4 you can use an IBM MQ application to create messages that are put on a queue manager topic `/*root*/mqtosfb/event/+`. The bridge subscribes to the topic, gets content from the messages, and uses it to publish event messages for a Salesforce platform event. For more information on platform events, see Delivering custom notifications with platform events in Salesforce developer documentation.

To enable the bridge to create event messages, you must provide two attributes additional to those at IBM MQ Version 9.0.2 that were used for subscribing to push topics and platform events:

- Create and add the name of the **`MQ Publication Error Queue`** in the bridge configuration attributes for Connection to Queue Manager.
- Set the **`Subscribe to MQ publications for platform events`** option to *Y*, in the bridge configuration attributes for defining the Behavior of bridge program.

You need to create a platform event in Salesforce and define the content fields before you can use the bridge to create event messages for that platform event. The platform event name and its contents determine how you need to format the IBM MQ message that is processed by the bridge. For example, if your Salesforce platform event **`Object name`** is *MQPlatformEvent1* and your two custom defined fields are text fields with the **`API name`** *MyText__c* and *Name__c*, then your IBM MQ message that is published on the `/`*`root`*`/mqtosfb/event/MQPlatformEvent1__e` topic must be a correctly formatted JSON, as follows:

```
{ "MyText__c" : "Some text here", "Name__c" : "Bob Smith" }
```

See step 7 on page 1438 to create your platform event in Salesforce or skip this step if you already have a platform event that you want to create event messages for. You have to format your IBM MQ message to match the fields that are set in your Salesforce platform event. Fields within the Salesforce platform event can be designated as optional or mandatory. For more information, see Platform Event Fields in the Salesforce developer documentation.

When the bridge is running, it subscribes to the designated IBM MQ topic.

- If you specify the **`At-most-once`** quality of service in the bridge configuration, the subscription that the bridge makes is non-durable. Any publications that are made by IBM MQ applications while the bridge is not running are not processed.
- If you specify the **`At-least-once`** quality of service in the bridge configuration, the subscription the bridge makes is durable. This means that the bridge can process publications that are made by IBM MQ applications while the bridge is not running. Durable subscriptions require a known subscription and client ID. The bridge uses *D_SUB_RUNMQSFB* as the subscription name and *runmqsfb_1* as the client ID.

If the bridge is used for subscribing to Salesforce push topics and platform events, and not for creating event messages, it attempts to delete the durable subscription, in case the configuration is changed, and the subscription is now orphaned.

You can remove durable subscriptions that the bridge creates as follows:

Use the IBM MQ Explorer.
Open the **subscriptions folder** for the queue manager that the bridge is using and look for the subscription name that ends in *:D_SUB_RUNMQSFB* where the topic string is `/sf/mqtosfb/event+`. Right-click the subscription name and click delete. If you get an error that indicates that the subscription is in use, your bridge might still be running. Stop the bridge and try to delete the subscription again.

Use `runmqsc` to find and delete the subscription.
Start the **`runmqsc`** interface and run `DISPLAY SUB (*)`. Look for the subscription name **`SUB`** ending in *:D_SUB_RUNMQSFB*. Issue the delete sub command and include the **`SUBID`** of the subscription you want to delete. For example, `DELETE SUB SUBID(414D5120514D312020202020202020205C589459987E8620)`

Stop, then start the bridge with the `At-most-once` quality of service.
If you started the bridge with the **`At-least-once`** quality of service `At-least-once delivery? (Y/N) :[Y]`, the subscription that is created is durable. To delete the subscription, change the quality of service to `At-least-once delivery? (Y/N) :[N]` in your configuration file and restart the bridge. The durable subscription is deleted and a non-durable subscription is created.

Procedure

1. Create and start a queue manager.
   a. Create a queue manager, for example PEQM1.
      ```
      crtmqm PEQM1
      ```
   b. Start your queue manager.
      ```
      strmqm PEQM1
      ```
2. Optional:

   Note: To use existing login and security Salesforce credentials and self-signed certificate, skip to step 4.
   Create a security token for your Salesforce account.
   a. Log in to your Salesforce account.
   b. Create or reset your security token by following the steps in the help article Salesforce help: Reset your security token.
3. Create a self-signed security certificate in Salesforce.
   a. Select **Security controls** from the **Administer** menu of your **Force.com Home** page, then **Certificate and Key Management**. The Certificate and Key Management page opens.
   b. Click **Create Self-Signed certificate**. The Certificates page opens.
   c. Enter a name for the certificate in the **Label** field, press `Tab`, then click **Save**. The Certificate and Key Detail information is displayed.
   d. Click **Back to list: Certificates and keys**.
   e. Click **Export to Keystore**.
   f. Enter a password for the keystore, then click **Export**.
   g. Save the exported keystore to your local file system.
4. Use the IBM Key Management GUI to open the keystore you exported from Salesforce and populate the signer certificates.
   a. Run the **`strmqikm`** command to open the IBM Key Management GUI. For more information, see Using runmqckm, runmqakm, and strmqikm to manage digital certificates.
   b. Click **Open a key database file** and browse to the location of the Salesforce keystore.
   c. Click **Open**, make sure to select **JKS** from the **Key database type** options, then click **OK**.
   d. Enter the password that you created for the keystore in step 3f, then click **OK**.
   e. Select **Signer Certificates** from the **Key database content** options.
   f. Click **Populate**.
   g. Select the **Verisign Inc.** check box from the **Add CA Certificates** list, then click **OK**.
5. Optional: Generate OAuth consumer key and secret by creating an app connection for IBM MQ Bridge to Salesforce in your Salesforce account. You need the **Consumer Key** and **Consumer Secret** codes when you are using the IBM MQ Bridge to Salesforce in production environments.
   a. Select **Create**, then **Apps** from the **Build** menu of your **Force.com Home** page. The Apps page opens.
   b. Click **New** from the **Connected Apps** section. The New Connected App page opens.
   c. Enter a name for your IBM MQ Bridge to Salesforce in the **Connected App Name**, for example **MQBridgeToSalesforce**.
   d. Enter the **API Name**. If you tab through to the next field, the **Connected App Name** is copied into the **API Name** name field.
   e. Enter your **Contact Email**.
   f. Select the **Enable OAuth Settings** option in the **API (Enable OAuth Settings)** section. Further options in that section are then presented.
   g. Add your **Callback URL**, for example `https://www.ibm.com`.

   h. Select the **Full access (full)** option from the **Available OAuth Scopes** list in the **Selected OAuth Scopes** subsection, then click **Add**, to add full access to the **Selected OAuth Scopes** list.
   i. Click **Save**.
   j. Click **Continue**.
   k. Take note of your **Consumer Key** and **Consumer Secret** codes.

6. Create the required synchronization and error queues on the queue manager.

```
cat /opt/mqm/mqsf/samp/mqsfbSyncQ.mqsc | runmqsc PEQM1
```

The synchronization queue maintains event state across application or queue manager restarts. The queue depth can be small as only a single message is expected on the queue. Only one instance of the bridge can run at a time against this queue, so the default options are set for exclusive access. The error queue must be created before you can use the bridge to create event messages for platform events. The error queue is used for messages that cannot successfully be processed by Salesforce. You must add the error queue name in the bridge configuration parameter section **`Connection to Queue Manager`** as shown in step 8a.

7. Optional: Create a platform event object in your Salesforce account.
   a. Select **Platform Events** from the **Develop** menu of your **Force.com Home** page, then click **New Platform Event**. The New Platform Event page opens.
   b. Complete the **Label** and **Plural Label** fields.
   c. Click **Save.** The **Platform Event Definition Detail** page opens.
   d. Define the **Custom Fields & Relationships**. For example, you might add two text fields with labels *MyText* and *Name* and set the **Data Type** field lengths to *Text(64)* and *Text(32)* respectively.

You created a platform event and defined **`Custom Fields and Relationships`** for it. Use your platform event *Platform Object name* or the *API name* as the IBM MQ topic to which you can put messages that you want the bridge to process. For example, you might use the **`AMQSPUBA`** sample to add the following JSON formatted message to the `/sf/mqtosfb/event/`*`Salesforce Platform Object Name`*`/`*`API name`* topic:

```
{ "MyText__c" : "Some text here", "Name__c" : "Bob Smith" }
```

You can run the **`AMQSPUBA`** sample to create messages after the bridge started. From the *`MQ installation location`*`/samp/bin` directory, issue the following command:

```
./amqspub /sf/mqtosfb/event/Salesforce Platform Object Name/API name PEQM1
```

At the prompt, enter the message in JSON format.

8. Create a configuration file with connection and security parameters for IBM MQ, Salesforce, and the IBM MQ Bridge to Salesforce behavior.

```
runmqsfb -o new_config.cfg
```

The existing values are shown inside the brackets. Press `Enter` to accept existing values, press `Space` then `Enter` to clear values, and, type, then `Enter` to add new values.

   a. Enter values for the connection to queue manager PEQM1: Minimum values that are needed for the connection are queue manager name, IBM MQ base topic root, error queue name, and channel name.

```
Connection to Queue Manager
---------------------------
Queue Manager or JNDI CF   : []PEQM1
MQ Base Topic              : []/sf
MQ Channel                 : []A channel you have defined or for example SYSTEM.DEF.SVRCONN
MQ Conname                 : []
MQ Publication Error Queue : [SYSTEM.SALESFORCE.ERRORQ]
MQ CCDT URL                : []
```

```
JNDI implementation class  : [com.sun.jndi.fscontext.RefFSContextFactory]
JNDI provider URL          : []
MQ Userid                  : []
MQ Password                : []
```

Note: If you are connecting locally, the channel name is not required. You don't have to provide the queue manager name and base topic in the configuration file as they can be included on the command line later, when you run the bridge.

b. Enter values for connection to Salesforce: Minimum values that are needed for the connection are Salesforce userid, password, security token, and login endpoint. In production environments, you can add the consumer key and secret for OAuth security.

```
Connection to Salesforce
------------------------
Salesforce Userid (reqd)   : []salesforce_login_email
Salesforce Password (reqd) : []salesforce_login_password
Security Token (reqd)      : []Security_Token
Login Endpoint             : [https://login.salesforce.com]
Consumer ID                : []
Consumer Secret Key        : []
```

c. Enter values for certificate stores for TLS connections: Minimum values that are needed for TLS connections are the path to the keystore for TLS certificates and keystore password. If no trusted store path or password is provided, the keystore and password parameters are used for the trusted store and password. If you are using TLS for your IBM MQ queue manager connection, you can use the same keystore.

```
Certificate stores for TLS connections
--------------------------------------
Personal keystore for TLS certificates : []path_to_keystore, for example: /var/mqm/qmgrs/PEQM1/ssl/key.jks
Keystore password          : []keystore_password
Trusted store for signer certificates : []
Trusted store password     : []
Use TLS for MQ connection  : [N]
```

d. Enter values to configure the behavior of the IBM MQ Bridge to Salesforce: You must change the **`Subscribe to MQ publications for platform events`** option from the default *N*, to *Y* , to use the bridge to create event messages. You also must specify the log file, in the configuration file or on the command line.

```
Behaviour of bridge program
---------------------------
PushTopic Names            : []
Platform Event Names       : []
MQ Monitoring Frequency    : [30]
At-least-once delivery? (Y/N) : [Y]
Subscribe to MQ publications for platform events? (Y/N) : [Y]
Publish control data with the payload? (Y/N) : [N]
Delay before starting to process events : [0]
Runtime logfile for copy of stdout/stderr : []
```

9. Optional: Create the IBM MQ service to control the execution of the program. Edit the sample `mqsfbService.mqsc` file to point to the newly created configuration file and make any other changes to the command parameters.

```
cat modified mqsfbService.mqsc | runmqsc PEQM1
```

10. V 9.0.1 Optional: Follow instructions in Getting started with the IBM MQ Console to set up the IBM MQ Console.

11. Optional: Add and configure widgets in your IBM MQ Console instance to view Salesforce data.
    a. Click **Add widget**. The new widget opens.
    b. Select **Charts**
    c. Click **Configure widget** icon in the title bar of the new widget.
    d. Optional: Enter a **Widget title**.

e. Select **Salesforce Bridge** from the **Resource to monitor**, **Source** drop-down menu.
f. Select **Bridge Status**, from the **Resource class**, drop-down menu.
g. Select **MQ-created Platform Events**, from the **Resource type**, drop-down menu.
h. Select **Total MQ-created Platform Events**, from the **Resource element**, drop-down menu.
i. Click **Save**.

You configured the IBM MQ Console for showing the total number of IBM MQ created platform events. When the bridge is running and you start putting messages on the `/sf/mqtosfb/event/`*`Salesforce Platform Object Name/API name`* topic, the widget shows the number of total message events that the bridge created.

Message format and error messages for the IBM MQ Bridge to Salesforce

V 9.0.4

Information on formatting of the messages that are processed by the IBM MQ Bridge to Salesforce.

An application puts a message to a specific queue manager topic, for example */root*`/mqtosfb/event/MQPlatformEvent1__e`. The bridge subscribes to the topic, gets content from the messages, and uses it to publish event messages for a Salesforce platform event.

You need to create a platform event in Salesforce and define the content fields before you can use the bridge to create event messages for that platform event. The platform event name and its contents determine how you need to format the IBM MQ message that is processed by the bridge. For example, if your Salesforce platform event **`Object name`** is *MQPlatformEvent1* and your two custom defined fields are text fields with the **`API name`** *MyText__c* and *Name__c*, then your IBM MQ message that is published on the */root*`/mqtosfb/event/MQPlatformEvent1__e` topic must be a correctly formatted JSON, as follows:

```
{ "MyText__c" : "Some text here", "Name__c" : "Bob Smith" }
```

The messages that are consumed and produced by the bridge are text (MQSTR) messages in JSON format. The input message is a simple JSON and programs can use string concatenation to generate it.

Error messages

Errors can be detected by the bridge, for example if the message is not in text format or by Salesforce, for example if the platform event name does not exist. If an error occurs in processing the input message, the message is moved to the bridge error queue along with the properties that describe the error. The error is also written to the *stderr* stream for the bridge.

Errors that are generated by Salesforce are JSON. The following are some errors that are caused by incorrectly formatted messages:

Bad platform event contents, status 400 Text

```
[{"message":"No such column 'Name__c' on sobject of type MQPlatformEvent2__e","errorCode":"INVALID_FIELD"}
```

Invalid platform event name, status 404 text

```
{"errorCode":"NOT_FOUND","message":"The requested resource does not exist"}
```

Bad JSON, status 400 text

```
{"errorCode":"NOT_FOUND","message":"The requested resource does not exist"}
```

Message is not JSON, status 400 text

```
[{"message":
  "Unexpected character ('h' (code 104)): expected a valid value (number, String, array, object, 'true', 'false' or 'null')
  "errorCode":"JSON_PARSER_ERROR"}
```

Not a text message (not sent to Salesforce)

```
Error: Publication on topic ' /sf/mqtosfb/event/MQPlatformEvent1' does not contain a text formatted message
```

Running the IBM MQ Bridge to Salesforce

Linux V 9.0.2

Run the IBM MQ Bridge to Salesforce to connect to Salesforce and IBM MQ. When connected, the bridge can create subscriptions to Salesforce topics and republish messages to the IBM MQ topic.

V 9.0.4 From IBM MQ Version 9.0.4 the bridge can also create event messages for Salesforce platform events.

Before you begin

You completed configuration steps in task:
- "Configuring the IBM MQ Bridge to Salesforce" on page 1431
- V 9.0.4 "Creating event messages for Salesforce platform events" on page 1435

.

About this task

Use the configuration file that you created in the previous task, to run the IBM MQ Bridge to Salesforce. If you have not included all the required parameters in your configuration file, make sure to include them in command line.

Procedure

1. Define the push topics or platform events in Salesforce that you want to subscribe to V 9.0.4 or the platform event that you want to create event messages for..
2. Start the IBM MQ Bridge to Salesforce to connect to Salesforce and your queue manager. If you are running the bridge to subscribe to Salesforce events, include the name of the push topic or platform event that you defined in step 1.

   ```
   runmqsfb -f new_config.cfg -r logFile -p PushtopicName -e eventName
   ```

   When the bridge is connected, the following messages are returned:

   At IBM MQ Version 9.0.2

   ```
   Successful connection to queue manager QM1
   Successful login to Salesforce at https://eu11.salesforce.com
   Ready to process events.
   ```

   V 9.0.4 At IBM MQ Version 9.0.4
   - If you are using the bridge to subscribe to Salesforce push topic and platform events:

     ```
     Successful connection to queue manager QM1
     Warning: Subscribing to MQ-created platform events is not enabled.
     Successful login to Salesforce at https://eu11.salesforce.com
     Ready to process events.
     ```
   - If you are using the bridge to create event messages for Salesforce platform events:

     ```
     Successful connection to queue manager QM1
     Successful login to Salesforce at https://eu11.salesforce.com
     Successful subscription to '/sf/mqtosfb/event/+' for MQ-created platform events
     Ready to process events.
     ```
3. Optional: Troubleshoot the connection to your queue manager and to Salesforce if the messages that are returned after you run the bridge indicate that a connection was not successful.

   a. Issue the command in debug mode with the debug option 1.

```
runmqsfb -f new_config.cfg -r logFile -p PushtopicName -e eventName -d 1
```

   The bridge steps through the connection set up and shows the processing messages in terse mode.

   b. Issue the command in debug mode with the debug option 2.

```
runmqsfb -f new_config.cfg -r logFile -p PushtopicName -e eventName -d 2
```

   The bridge steps through the connection set up and shows the processing messages in verbose mode. Full output is written to your log file.

4. Generate events by using the Salesforce interface to modify records in the database.
5. Go to the IBM MQ Console to see changes to push topics appear in the widget you configured in the previous task.

What to do next

Use the *MQSFB_EXTRA_JAVA_OPTIONS* variable to pass in JVM properties, for example, to enable IBM MQ tracing. For more information, see Tracing the IBM MQ Bridge to Salesforce.

Related information:

runmqsfb (run IBM MQ Bridge to Salesforce)

Monitoring the IBM MQ Bridge to Salesforce

Configuring IBM MQ for use with blockchain

Linux V 9.0.4 MQ Adv.

Set up and run the IBM MQ Bridge to blockchain to securely connect an IBM MQ Advanced queue manager and IBM Blockchain. Use the bridge to asynchronously connect to, look up and update the state of a resource in your blockchain, by using a messaging application that connects to your IBM MQ Advanced queue manager.

Before you begin

- IBM MQ Bridge to blockchain is available for connecting to IBM MQ Advanced queue managers only.
- The queue manager must be at the same command level as the bridge, for example Version 9.0.4.
- IBM MQ Bridge to blockchain is supported for use with your blockchain network that is based on Hyperledger Fabric v1.0 architecture.

About this task

Blockchain is a shared, distributed, digital ledger that consists of a chain of blocks that represent agreed upon transactions between peers in a network. Each block in the chain is linked to the previous block, and so on, back to the first transaction.

IBM Blockchain is built on Hyperledger Fabric and you can develop with it locally with Docker or in a container cluster in IBM Cloud (formerly Bluemix). You can also activate and use your IBM Blockchain network in production, to build, and govern a business network with high levels of security, privacy, and performance. For more information, see the IBM Blockchain Platform.

Hyperledger Fabric is an open source, enterprise blockchain framework that is developed collaboratively by members of the Hyperledger Project, including IBM as the initial code contributor. Hyperledger Project, or Hyperledger, is a Linux Foundation open source, global, collaborative initiative to advance cross-industry blockchain technologies. For more information, see IBM Blockchain, Hyperledger Projects, and Hyperledger Fabric.

If you are already using IBM MQ Advanced and IBM Blockchain, you can use the IBM MQ Bridge to blockchain to send simple queries, updates, and receive replies from your blockchain network. In this way, you can integrate your on-premises IBM software with a cloud blockchain service.

A brief overview of the bridge operating process can be seen in Figure 1. A user application puts a JSON formatted message on the input/request queue on the IBM MQ Advanced queue manager. The bridge connects to the queue manager, gets the message from the input/request queue, checks that the JSON is correctly formatted, then issues the query or an update to the blockchain. The data that is returned by the blockchain is parsed by the bridge and placed on the reply queue, as defined in the original IBM MQ request message. The user application can connect to the queue manager, get the response message from the reply queue, and use the information.

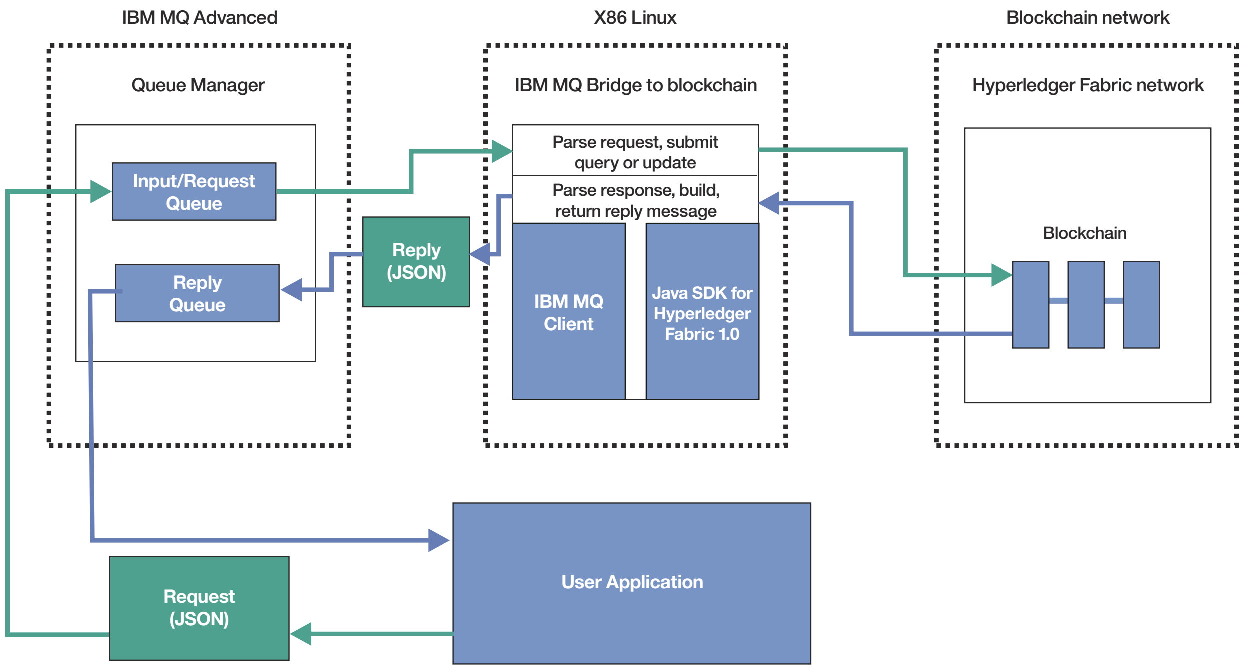

IBM MQ Advanced
Queue Manager
Input/Request Queue
Reply Queue
X86 Linux
IBM MQ Bridge to blockchain
Parse request, submit query or update
Parse response, build, return reply message
IBM MQ Client
Java SDK for Hyperledger Fabric 1.0
Reply (JSON)
Blockchain network
Hyperledger Fabric network
Blockchain
User Application
Request (JSON)

Figure 175. IBM MQ Bridge to blockchain

You can configure the IBM MQ Bridge to blockchain to connect to a blockchain network as a participant, or peer. When the bridge is running, a messaging application requests the bridge to drive chaincode routines that query or update the state of the resource and return the results as a response, to the messaging application.

Procedure

1. Create and start a queue manager, or start an existing queue manager that you want to use with your IBM MQ Bridge to blockchain. Create queue manager:

   ```
   crtmqm adv_qmgr_name
   ```

   Start queue manager:

   ```
   strmqm adv_qmgr_name
   ```

2. Create the queues for the bridge that are defined in the **DefineQ.mqsc** script. Sample bridge queue definitions are provided for the default named queues that are used for:
   - User credentials, for example `SYSTEM.BLOCKCHAIN.IDENTITY.QUEUE`
   - Message input to the bridge, for example `APPL1.BLOCKCHAIN.INPUT.QUEUE`

- Replies from blockchain, for example `APPL1.BLOCKCHAIN.REPLY.QUEUE`

From the `/opt/mqm/mqbc/samp` directory, issue the following command:

```
runmqsc adv_qmgr_name < ./DefineQ.mqsc
```

Different applications can use the same input queue but you can specify multiple reply queues, one for each of your applications. You do not have to use defined reply queues. If you want to use dynamic queues for replies, you must consider their security configuration.

Results

You created the queues that the bridge requires for processing messages from IBM MQ and your blockchain network.

What to do next

Use your IBM MQ Advanced queue manager information and the credentials from your blockchain network to create a configuration file for the IBM MQ Bridge to blockchain.

Creating the configuration file for the IBM MQ Bridge to blockchain

V 9.0.4

Enter your queue manager and your blockchain network parameters to create the configuration file for the IBM MQ Bridge to blockchain to connect to your IBM MQ and IBM Blockchain networks.

Before you begin

- You created and configured your blockchain network.
- You have the credentials file from your blockchain network.
- You installed the IBM MQ Bridge to blockchain, on your x86 Linux environment.
- You started your IBM MQ Advanced queue manager.

About this task

This task takes you through the minimal setup that is needed to create the IBM MQ Bridge to blockchain configuration file and successfully connect to your IBM Blockchain and IBM MQ networks.

You can use the bridge to connect to blockchain networks that are based on Hyperledger Fabric v1.0 architecture. To use the bridge, you need configuration information from your blockchain network. In each step in this task you can find example configuration details that are based on two differently configured blockchain networks:

- Hyperledger Fabric network that runs in Docker. For more information, see Getting started with Hyperledger Fabric, Writing your first application, and "Example Hyperledger Fabric network credentials file" on page 1446.
- Hyperledger Fabric network that runs in a Kubernetes cluster in IBM Cloud (formerly Bluemix). For more information, see Develop in a cloud sandbox on IBM Blockchain Platform, and "Example Kubernetes container cluster network configuration file" on page 1449.

For more information on the meaning and options for all the IBM MQ Bridge to blockchain parameters, see the runmqbcb command. You must consider your own security requirements and customize the parameters appropriate to your deployment.

Procedure

1. Run the bridge to create a configuration file. You need the parameters from your blockchain network credentials file and from your IBM MQ Advanced queue manager.

```
runmqbcb -o config_file_name.cfg
```

As the following example illustrates, the existing values are shown inside the brackets. Press `Enter` to accept existing values, press `Space` then `Enter` to clear values, and type inside the brackets then press `Enter` to add new values. You can separate lists of values (such as peers) by commas, or by entering each value on a new line. A blank line ends the list.

Note: You cannot edit the existing values. You can keep, replace, or clear them.

2. Enter values for the connection to your IBM MQ Advanced queue manager. Minimum values that are needed for the connection are the queue manager name, the names of the bridge input and identity queues that you defined. For connections to remote queue managers, you also need **`MQ Channel`** and **`MQ Conname`** (host address and port where the queue manager is running). To use TLS for connecting to IBM MQ in step 6 on page 1446, you must use JNDI or CCDT and specify **`MQ CCDT URL`** or **`JNDI implementation class`** and **`JNDI provider URL`** accordingly.

```
Connection to Queue Manager
---------------------------
Queue Manager                       : [adv_qmgr_name]
Bridge Input Queue                  : [APPL1.BLOCKCHAIN.INPUT.QUEUE]
Bridge User Identity Queue          : [SYSTEM.BLOCKCHAIN.IDENTITY.QUEUE]
MQ Channel                          : []
MQ Conname                          : []
MQ CCDT URL                         : []
JNDI implementation class           : []
JNDI provider URL                   : []
MQ Userid                           : []
MQ Password                         : []
```

3. Enter the login details for the certificate authority for your blockchain network. The default values for your local Hyperledger Fabric and Kubernetes cluster examples are *admin* for **`Userid`** and *adminpw* for **`Enrollment Secret`**. If you changed these values for your blockchain network, ensure that you use the correct values to configure the bridge.

```
Blockchain - User Identification
--------------------------------
Blockchain Userid                   : []admin
Enrollment Secret                   : []******
```

4. Enter the membership service provider id (**`MSPid`**) that governs membership and identity rules for your blockchain network. From your credentials file, provide the **`msp_id`** parameter for the **`Organisation Name`** and **`Organisation MSPId`**. From the "Example Hyperledger Fabric network credentials file" on page 1446, use the **`CORE_PEER_LOCALMSPID`** value from the `peer` section of the file. From the "Example Kubernetes container cluster network configuration file" on page 1449, use the **`mspID`** value.

```
Blockchain - Organisation Identification
----------------------------------------
Organisation Name                   : []Org1MSP
Organisation MSPId                  : []Org1MSP
```

5. Enter your blockchain network server location values: From your "Example Hyperledger Fabric network credentials file" on page 1446, provide the names and server:port locations for certificate authority, peer, and orderer elements.

```
Blockchain server locations
---------------------------
Certificate Authority servers       : [ca.example.com Docker_container_host:7054] (for example ca.example.com localh
Peer servers                        : [peer0 localhost:7051]
Orderer servers                     : [orderer0 localhost:7050]
Peer Event servers                  : [peer0 localhost:7053]
Location of PEM file for Blockchain certificate : []
```

From your "Example Kubernetes container cluster network configuration file" on page 1449, provide the names and server:port locations for certificate authority, peer, and orderer elements.

```
Blockchain server locations
---------------------------
Certificate Authority servers       : [CA1                 your_blockchain_network_public_ip_address:30000] (for examp
Peer servers                        : [blockchain-org1peer1 your_blockchain_network_public_ip_address:30110]
Orderer servers                     : [blockchain-orderer   your_blockchain_network_public_ip_address:31010]
Peer Event servers                  : [blockchain-org1peer1 your_blockchain_network_public_ip_address:30111]
Location of PEM file for Blockchain certificate : []
```

6. Enter certificate stores values for TLS connections. The bridge acts as an IBM MQ Java client that is connecting to a queue manager, which means that it can be configured to use TLS security to connect securely in the same way as any other IBM MQ Java client. Configuration of TLS connection details is exposed only after you specify JNDI or CCDT information in step 2 on page 1445.

```
Certificate stores for TLS connections
-------------------------------------
Personal keystore                   : []
Keystore password                   : []
Trusted store for signer certs      : []
Trusted store password              : []
Use TLS for MQ connection           : [N]
Timeout for Blockchain operations   : [12]
```

7. Enter the location for the log file for the IBM MQ Bridge to blockchain. You must specify the log file name and location, in the configuration file or on the command line.

```
Behavior of bridge program
--------------------------
Runtime logfile for copy of stdout/stderr : [/var/mqm/errors/runmqbcb.log]
Done.
```

Results

You created the configuration file that the IBM MQ Bridge to blockchain uses to connect to your IBM Blockchain network and to your IBM MQ Advanced queue manager.

What to do next

Work through the steps for “Running the IBM MQ Bridge to blockchain” on page 1449.

Example Hyperledger Fabric network credentials file

V 9.0.4

Contents of the `.yml` file from your locally instantiated Hyperledger Fabric blockchain network running in Docker, that you can use to configure your IBM MQ Bridge to blockchain.

After you have worked through the Getting started with Hyperledger Fabric tutorial, understood What's happening behind the scenes, and have launched your network by using one of the Hyperledger Fabric samples, you should have the following configuration file in your `/blockchain/fabric-samples/basic-network` folder.

If you want to connect to your blockchain network, you must use the configuration details from this file when you are “Creating the configuration file for the IBM MQ Bridge to blockchain” on page 1444.

```
#
# Copyright IBM Corp All Rights Reserved
#
# SPDX-License-Identifier: Apache-2.0
#
version: '2'

networks:
  basic:

services:
  ca.example.com:
    image: hyperledger/fabric-ca
    environment:
      - FABRIC_CA_HOME=/etc/hyperledger/fabric-ca-server
      - FABRIC_CA_SERVER_CA_NAME=ca.example.com
    ports:
      - "7054:7054"
    command: sh -c 'fabric-ca-server start --ca.certfile /etc/hyperledger/fabric-ca-server-config/ca.org1.example.com-cer
/etc/hyperledger/fabric-ca-server-config/f329434b83a06f32f17a300fefd841cfd16ff58f3185fb744aae047207b01a9e_sk -b admin:adm
    volumes:
      - ./crypto-config/peerOrganizations/org1.example.com/ca/:/etc/hyperledger/fabric-ca-server-config
    container_name: ca.example.com
    networks:
      - basic

  orderer.example.com:
    container_name: orderer.example.com
    image: hyperledger/fabric-orderer
    environment:
      - ORDERER_GENERAL_LOGLEVEL=debug
      - ORDERER_GENERAL_LISTENADDRESS=0.0.0.0
      - ORDERER_GENERAL_GENESISMETHOD=file
      - ORDERER_GENERAL_GENESISFILE=/etc/hyperledger/configtx/genesis.block
      - ORDERER_GENERAL_LOCALMSPID=OrdererMSP
      - ORDERER_GENERAL_LOCALMSPDIR=/etc/hyperledger/msp/orderer/msp
    working_dir: /opt/gopath/src/github.com/hyperledger/fabric/orderer
    command: orderer
    ports:
      - 7050:7050
    volumes:
        - ./config/:/etc/hyperledger/configtx
        - ./crypto-config/ordererOrganizations/example.com/orderers/orderer.example.com/:/etc/hyperledger/msp/orderer
        - ./crypto-config/peerOrganizations/org1.example.com/peers/peer0.org1.example.com/:/etc/hyperledger/msp/peerOrg1
    networks:
      - basic

  peer0.org1.example.com:
    container_name: peer0.org1.example.com
    image: hyperledger/fabric-peer
    environment:
      - CORE_VM_ENDPOINT=unix:///host/var/run/docker.sock
      - CORE_PEER_ID=peer0.org1.example.com
      - CORE_LOGGING_PEER=debug
      - CORE_CHAINCODE_LOGGING_LEVEL=DEBUG
      - CORE_PEER_LOCALMSPID=Org1MSP
      - CORE_PEER_MSPCONFIGPATH=/etc/hyperledger/msp/peer/
      - CORE_PEER_ADDRESS=peer0.org1.example.com:7051
      # # the following setting starts chaincode containers on the same
      # # bridge network as the peers
      # # https://docs.docker.com/compose/networking/
      - CORE_VM_DOCKER_HOSTCONFIG_NETWORKMODE=${COMPOSE_PROJECT_NAME}_basic
      - CORE_LEDGER_STATE_STATEDATABASE=CouchDB
      - CORE_LEDGER_STATE_COUCHDBCONFIG_COUCHDBADDRESS=couchdb:5984
      # The CORE_LEDGER_STATE_COUCHDBCONFIG_USERNAME and CORE_LEDGER_STATE_COUCHDBCONFIG_PASSWORD
      # provide the credentials for ledger to connect to CouchDB.  The username and password must
      # match the username and password set for the associated CouchDB.
      - CORE_LEDGER_STATE_COUCHDBCONFIG_USERNAME=
      - CORE_LEDGER_STATE_COUCHDBCONFIG_PASSWORD=
    working_dir: /opt/gopath/src/github.com/hyperledger/fabric
    command: peer node start
    # command: peer node start --peer-chaincodedev=true
    ports:
      - 7051:7051
      - 7053:7053
    volumes:
        - /var/run/:/host/var/run/
        - ./crypto-config/peerOrganizations/org1.example.com/peers/peer0.org1.example.com/msp:/etc/hyperledger/msp/peer
        - ./crypto-config/peerOrganizations/org1.example.com/users:/etc/hyperledger/msp/users
        - ./config:/etc/hyperledger/configtx
    depends_on:
      - orderer.example.com
      - couchdb
    networks:
      - basic

  couchdb:
    container_name: couchdb
    image: hyperledger/fabric-couchdb
    # Populate the COUCHDB_USER and COUCHDB_PASSWORD to set an admin user and password
    # for CouchDB.  This will prevent CouchDB from operating in an "Admin Party" mode.
    environment:
      - COUCHDB_USER=
      - COUCHDB_PASSWORD=
    ports:
      - 5984:5984
    networks:
      - basic

  cli:
    container_name: cli
    image: hyperledger/fabric-tools
    tty: true
    environment:
      - GOPATH=/opt/gopath
      - CORE_VM_ENDPOINT=unix:///host/var/run/docker.sock
      - CORE_LOGGING_LEVEL=DEBUG
      - CORE_PEER_ID=cli
      - CORE_PEER_ADDRESS=peer0.org1.example.com:7051
      - CORE_PEER_LOCALMSPID=Org1MSP
      -
CORE_PEER_MSPCONFIGPATH=/opt/gopath/src/github.com/hyperledger/fabric/peer/crypto/peerOrganizations/org1.example.com/users/
      - CORE_CHAINCODE_KEEPALIVE=10
    working_dir: /opt/gopath/src/github.com/hyperledger/fabric/peer
    command: /bin/bash
    volumes:
        - /var/run/:/host/var/run/
        - ./../chaincode/:/opt/gopath/src/github.com/
        - ./crypto-config:/opt/gopath/src/github.com/hyperledger/fabric/peer/crypto/
    networks:
        - basic
    #depends_on:
    #  - orderer.example.com
    #  - peer0.org1.example.com
    #  - couchdb
```

Example Kubernetes container cluster network configuration file

V 9.0.4

Contents of the configuration file from your Hyperledger Fabric blockchain network that is running in a Kubernetes cluster in IBM Cloud (formerly Bluemix), that you can use to configure your IBM MQ Bridge to blockchain.

After you have worked through the IBM Blockchain Prepare and setup, Simple install, and Interacting with your blockchain tutorials, you should have a JSON file in your connection profile folder.

If you want to connect to your blockchain network, you must use the configuration details from this file when you are "Creating the configuration file for the IBM MQ Bridge to blockchain" on page 1444.

```
{
     "name": "ibm-bc-org1",
     "description": "Connection profile for IBM Blockchain Platform",
     "type": "hlfv1",
     "orderers": [
         {
             "url": "grpc://INSERT_PUBLIC_IP:31010"
         }
     ],
     "ca": {
         "url": "http://INSERT_PUBLIC_IP:30000",
         "name": "CA1"
     },
     "peers": [
         {
             "requestURL": "grpc://INSERT_PUBLIC_IP:30110",
             "eventURL": "grpc://INSERT_PUBLIC_IP:30111"
         }
     ],
     "keyValStore": "INSERT_CREDENTIALS_PATH",
     "channel": "channel1",
     "mspID": "Org1MSP",
     "timeout": 300
 }
```

Running the IBM MQ Bridge to blockchain

V 9.0.4

Run the IBM MQ Bridge to blockchain to connect to IBM Blockchain and IBM MQ. When connected, the bridge is ready to process query and update messages, send them to your blockchain network, and receive and process the replies.

About this task

Use the configuration file that you created in the previous task, to run the IBM MQ Bridge to blockchain.

Procedure

1. Start the IBM MQ Advanced queue manager that you want to use with the bridge.
2. Start the IBM MQ Bridge to blockchain to connect to your blockchain network and your IBM MQ Advanced queue manager. Run the bridge command.

   ```
   runmqbcb -f /config_file_location/config_file_name.cfg -r /log_file_location/logFile.log
   ```

   When the bridge is connected, output similar to the following is returned:

```
Fri Oct 06 06:32:11 PDT 2017 IBM MQ Bridge to Blockchain
5724-H72 (C) Copyright IBM Corp. 2017

Fri Oct 06 06:32:17 PDT 2017 Ready to process input messages.
```

3. Optional: Troubleshoot connections to your IBM MQ Advanced queue manager and to your blockchain network, if the messages that are returned after you run the bridge indicate that a connection is not successful.
   a. Issue the command in debug mode with the debug option 1.
   ```
   runmqbcb -f /config_file_location/config_file_name.cfg -r /log_file_location/logFile.log -d 1
   ```
   The bridge steps through the connection set up and shows the processing messages in terse mode.
   b. Issue the command in debug mode with the debug option 2.
   ```
   runmqbcb -f /config_file_location/config_file_name.cfg -r /log_file_location/logFile.log -d 2
   ```
   The bridge steps through the connection set up and shows the processing messages in verbose mode. Full output is written to your log file.

Results

You have started the IBM MQ Bridge to blockchain and connected to your queue manager and blockchain network.

What to do next

- Follow the steps in "Running the IBM MQ Bridge to blockchain client sample" on page 1609 to format and send a query or update message to your blockchain network.
- Use the *MQBCB_EXTRA_JAVA_OPTIONS* variable to pass in JVM properties, for example to enable IBM MQ tracing. For more information, see Tracing the IBM MQ Bridge to blockchain.

Message formats for the IBM MQ Bridge to blockchain

V 9.0.4

Information on formatting of the messages that are sent and received by the IBM MQ Bridge to blockchain.

An application requests that the IBM MQ Bridge to blockchain performs a query or update of information that is held on the blockchain. The application does this by placing a request message on the bridge request queue. The results of the query or the update are formatted by the bridge into a reply message. The bridge uses information that is contained in the **ReplyToQ** and **ReplyToQMgr** fields from the MQMD of the request message as the destination for the reply message.

The messages that are consumed and produced by the bridge are text (MQSTR) messages in JSON format. The input message is a simple JSON and programs can use string concatenation to generate it. All the fields except **args** are required, the argument list for that field requires knowledge of the functions of the stored chaincode.

Request Message Format

Input message format:

```
{ "function": functionName,
    "channel" : chainName,
    "chaincodeName" : codeName,
    "args" : [argument list]
}
```

For the local hyperledger network example with the working Fabcar sample.

- To use the query message that calls the `queryAllCars` function in the fabcar chaincode that returns a list of JSON objects that represent the car details that are held in the blockchain, format the message as follows:

  ```
  { "function": "queryAllCars",
      "channel":"mychannel",
      "chaincodeName": "fabcar",
      "args":[]
  }
  ```

  Example reply:

  ```
  {
     "statusCode": 200,
     "statusType": "SUCCESS",
     "message": "OK",
     "data": [
    {"Record":{"owner":"Tomoko","colour":"blue","model":"Prius","make":"Toyota"},"Key":"CAR0"},
    {"Record":{"owner":"Brad","colour":"red","model":"Mustang","make":"Ford"},"Key":"CAR1"},
    {"Record":{"owner":"Jin Soo","colour":"green","model":"Tucson","make":"Hyundai"},"Key":"CAR2"},
    {"Record":{"owner":"Max","colour":"yellow","model":"Passat","make":"Volkswagen"},"Key":"CAR3"},
    {"Record":{"owner":"Adriana","colour":"black","model":"S","make":"Tesla"},"Key":"CAR4"},
    {"Record":{"owner":"Michel","colour":"purple","model":"205","make":"Peugeot"},"Key":"CAR5"},
    {"Record":{"owner":"Aarav","colour":"white","model":"S22L","make":"Chery"},"Key":"CAR6"},
    {"Record":{"owner":"Pari","colour":"violet","model":"Punto","make":"Fiat"},"Key":"CAR7"},
    {"Record":{"owner":"Valeria","colour":"indigo","model":"Nano","make":"Tata"},"Key":"CAR8"},
    {"Record":{"owner":"Shotaro","colour":"brown","model":"Barina","make":"Holden"},"Key":"CAR9"}
  ]}
  ```

  The reply message contains all the car records that are currently held in the blockchain.
- To use the update message that calls the `createCar` function in the fabcar example chaincode that creates a new car entry in the blockchain ledger, format the message as follows:

  ```
  { "function":"createCar",
    "channel":"mychannel",
    "chaincodeName":"fabcar",
    "args":["CAR10", "Ford", "Mustang GT", "Blue", "Bob"]
  }
  ```

  Example reply:

  ```
  {
     "statusCode": 200,
     "statusType": "SUCCESS",
     "message": "OK",
     "data": ""
  }
  ```

  To check that the new car entry is added to the blockchain, you can use the initial message again that returns all the cars.

For the Kubernetes cluster network example with the working example02 demo.

- To use the query message that calls the `query` function in the example02 chaincode that returns the value for entity *"a"* within the blockchain ledger, format the message as follows:

  ```
  { "function":"query",
    "channel":"channel1",
    "chaincodeName":"example02",
    "args":["a"]
  }
  ```

  Example reply:

```
{
   "statusCode": 200,
   "statusType": "SUCCESS",
   "message": "OK",
   "data": "100"
}
```

- To use the message that calls the invoke function example02 chaincode that decrements the entity that is specified in the first argument and increments the entity that is specified in the second argument by the value that is specified in the third argument, format the message as follows:

  ```
  { "function":"invoke",
    "channel":"channel1",
    "chaincodeName":"example02",
    "args":["a", "b", "10"]
  }
  ```

  The values are as follows:
  - Before: a=100, b=200
  - After: a=90, b=210

  Example reply:

  ```
  {
     "statusCode": 200,
     "statusType": "SUCCESS",
     "message": "OK",
     "data": ""
  }
  ```

  To check the new values, submit a new message query message to look for values of **"a"** and **"b"**.

Reply Message Format

Response messages have their correlation ID set to the message ID of the inbound message. Any user-defined properties are copied from the input to the output messages. The user ID in the reply is set to the originator's user ID through the set-identity context.

An example of successful processing:

```
{ "data": "500", "message": "OK", "statusCode": 200, "statusType": "SUCCESS" }
```

The response data in this message is whatever is generated from the chaincode response (bytes converted to a UTF-8 string).

All error responses have the same fields, regardless of whether they are generated by the bridge itself, from the calls to blockchain, or from the chaincode invocation. For example:

- Bad channel name

  ```
  {
     "message": "Bad newest block expected status 200 got 404, Chain myUnknownChannel",
     "statusCode": 404,
     "statusType": "FAILURE"
  }
  ```
- Bad JSON input message

  ```
  {
     "message": "Error: Cannot parse message contents.",
     "statusCode": 2110,
     "statusType": "FAILURE"
  }
  ```
- Incorrect parameters to chaincode

```
{
   "message": "Sending proposal to fabric-peer-1a failed because of gRPC failure=Status{code=UNKNOWN, description={\"E
   "statusCode": 500,
   "statusType": "FAILURE"
}
```

Applications can tell whether the request succeeded or failed by either looking at the **`statusType`** string, or from the existence of the data field. When there is an error in processing the input message, and the bridge does not send it to blockchain, the value that is returned from the bridge is an MQRC value, usually **`MQRC_FORMAT_ERROR`**.

Running the IBM MQ Bridge to blockchain client sample

V 9.0.4

You can use the JMS client sample that is provided with the IBM MQ Bridge to blockchain, to put a message on the input queue that the blockchain bridge is checking and see the reply that is received.

Before you begin

Your IBM MQ Bridge to blockchain is running and is connected to your IBM MQ Advanced queue manager and your blockchain network, and is ready to process input messages.

About this task

Find the JMS sample application in the `samp` directory of the IBM MQ Bridge to blockchain.

Procedure

1. Edit the client sample Java source file. Follow the instructions in the sample to configure it to match your IBM MQ environment and your blockchain network. The following code from the sample defines the JSON request message to send to the bridge:

```
 // Create the JSON request message.
 // Modify "query", "exampleBlockchainChannelName", and "exampleChaincodeName" to
 // match your deployed blockchain chaincode.
 // The "operation" field is optional, but recommended. It should be set to QUERY
 // or UPDATE to match what the chaincode is going to do.

JSONObject inputMsg = new JSONObject();
 inputMsg.put("operation", "QUERY");

inputMsg.put("function", "query");
 inputMsg.put("channel", "exampleBlockchainChannelName");
 inputMsg.put("chaincodeName","exampleChaincodeName");


// Create the JSON arguments for the request message.
 // Modify "a" to match your deployed blockchain chaincode
 // requirements, and add further arguments as necessary

JSONArray myArgs = new JSONArray();
 myArgs.add("a");
 inputMsg.put("args", myArgs);


 TextMessage message = session.createTextMessage(inputMsg.serialize());
 message.setJMSReplyTo(replyToQueue);
```

2. Compile the sample. Point to the IBM MQ client classes and `JSON4j.jar` file that is shipped in the bridge directory.

```
javac -cp $MQ_JAVA_INSTALL_PATH/lib/*:../prereqs/JSON4J.jar SimpleBCBClient.java
```

3. Run the compiled class.

```
java -cp $MQ_JAVA_INSTALL_PATH/lib/*:../prereqs/JSON4J.jar:. SimpleBCBClient
```

```
Starting Simple MQ Blockchain Bridge Client
Created the message. Starting the connection
Sent message:

 JMSMessage class: jms_text
  JMSType:          null
  JMSDeliveryMode:  2
  JMSDeliveryDelay: 0
  JMSDeliveryTime:  1508427559117
  JMSExpiration:    0
  JMSPriority:      4
  JMSMessageID:     ID:414d5120424342514d202020202020209063e859ea36aa24
  JMSTimestamp:     1508427559117
  JMSCorrelationID: null
  JMSDestination:   queue:///APPL1.BLOCKCHAIN.INPUT.QUEUE
  JMSReplyTo:       queue:///APPL1.BLOCKCHAIN.REPLY.QUEUE
  JMSRedelivered:   false
    JMSXAppID: java
   JMSXDeliveryCount: 0
    JMSXUserID: USER1
   JMS_IBM_PutApplType: 6
    JMS_IBM_PutDate: 20171019
    JMS_IBM_PutTime: 15391912
{"args":["a"],"function":"query","channel":"exampleBlockchainChannelName","operation":"QUERY","chaincodeName":"exampleCh
```

Response message:

```
  JMSMessage class: jms_text
  JMSType:          null
  JMSDeliveryMode:  1
  JMSDeliveryDelay: 0
  JMSDeliveryTime:  0
  JMSExpiration:    0
  JMSPriority:      4
  JMSMessageID:     ID:c3e2d840e2e2f0f840404040404040d2afa27229838af2
  JMSTimestamp:     1497439784000
  JMSCorrelationID: ID:414d5120424342514d202020202020209063e859ea36aa24  *(JMSMessageID of the input message)
  JMSDestination:   null
  JMSReplyTo:       null
  JMSRedelivered:   false
    JMSXAppID: java
    JMSXDeliveryCount: 1
    JMSXUserID: USER1
    JMS_IBM_Character_Set: UTF-8
    JMS_IBM_Encoding: 273
    JMS_IBM_Format: MQSTR
    JMS_IBM_MsgType: 8
    JMS_IBM_PutApplType: 2
    JMS_IBM_PutDate: 20171019
    JMS_IBM_PutTime: 15392014
{
   "data": "20",
   "message": "OK",
   "statusCode": 200,
   "statusType": "SUCCESS"
}
Response text:
{
   "data": "20",
   "message": "OK",
   "statusCode": 200,
   "statusType": "SUCCESS"
}
SUCCESS
```

If the client receives a timeout error waiting for the response, check that the bridge is running.

Configuring queue managers on z/OS

z/OS

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

Before you begin

Before you configure IBM MQ, read about the IBM MQ for z/OS concepts in IBM MQ for z/OS concepts.

z/OS Read about how to plan your IBM MQ for z/OS environment in Planning your IBM MQ environment on z/OS.

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 tasks:

"Creating and managing queue managers on Multiplatforms" on page 833
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.

"Configuring distributed queuing" on page 961
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 server and client" on page 844
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 information:

IBM MQ for z/OS concepts

Securing

z/OS Administering IBM MQ for z/OS

Planning

z/OS Issuing commands

z/OS The IBM MQ for z/OS utilities

Preparing to customize queue managers on z/OS

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 IBM MQ for z/OS V9.0 Long Term Support Release 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 the System Modification Program Extended (SMP/E).

When you have installed IBM MQ, you must carry out a number of tasks before you can make it available to users. See the following sections for a description of these tasks:

- "Customizing IBM MQ for z/OS" on page 1459
- "Testing a queue manager on z/OS" on page 1521
- 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 the Java Message Service (JMS) to connect to IBM MQ for z/OS or if you want to build and run HTTP applications which use HTTP to connect to IBM MQ for z/OS.

V 9.0.1 **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 Web Console, or the REST API to 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 136 on page 1457 shows the libraries that might exist after you have installed IBM MQ.

Table 136. IBM MQ libraries that exist after installation

Name	Description
thlqual.SCSQANLC	Contains the load modules for the Simplified Chinese version of IBM MQ.
thlqual.SCSQANLE	Contains the load modules for the U.S. English (mixed case) version of IBM MQ.
thlqual.SCSQANLF	Contains the load modules for the French version of IBM MQ.
thlqual.SCSQANLK	Contains the load modules for the Japanese version of IBM MQ.
thlqual.SCSQANLU	Contains the load modules for the U.S. English (uppercase) version of IBM MQ.
thlqual.SCSQASMS	Contains source for assembler sample programs.
thlqual.SCSQAUTH	The main repository for all IBM MQ product load modules; it also contains the default parameter module, CSQZPARM. This library must be APF-authorized and in PDS-E format.
thlqual.SCSQCICS	Contains extra load modules that must be included in the CICS DFHRPL concatenation. This library must be APF-authorized and in PDS-E format.
thlqual.SCSQCLST	Contains CLISTs used by the sample programs.
thlqual.SCSQCOBC	Contains COBOL copybooks, including copybooks required for the sample programs.
thlqual.SCSQCOBS	Contains source for COBOL sample programs.
thlqual.SCSQCPPS	Contains source for C++ sample programs.
thlqual.SCSQC37S	Contains source for C sample programs.
thlqual.SCSQC370	Contains C headers, including headers required for the sample programs.
thlqual.SCSQDEFS	Contains side definitions for C++ and the Db2 DBRMs for shared queuing.
thlqual.SCSQEXEC	Contains REXX executable files to be included in the SYSEXEC or SYSPROC concatenation if you are using the IBM MQ operations and control panels.
thlqual.SCSQHPPS	Contains header files for C++.
thlqual.SCSQINST	Contains JCL for installation jobs.
thlqual.SCSQLINK	Early code library. Contains the load modules that are loaded at system initial program load (IPL). The library must be APF-authorized.
thlqual.SCSQLOAD	Load library. Contains load modules for non-APF code, user exits, utilities, samples, installation verification programs, and adapter stubs. The library does not need to be APF-authorized and does not need to be in the link list. This library must be in PDS-E format.
thlqual.SCSQMACS	Contains Assembler macros including: sample macros, product macros, and system parameter macros.
thlqual.SCSQMAPS	Contains CICS mapsets used by sample programs.
thlqual.SCSQMSGC	Contains ISPF messages to be included in the ISPMLIB concatenation if you are using the Simplified Chinese language feature for the IBM MQ operations and control panels.
thlqual.SCSQMSGE	Contains ISPF messages to be included in the ISPMLIB concatenation if you are using the U.S. English (mixed case) language feature for the IBM MQ operations and control panels.

Table 136. IBM MQ libraries that exist after installation (continued)

Name	Description
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.
thlqual.SCSQPNLU	Contains ISPF panels to be included in the ISPPLIB concatenation if you are using the U.S. English (uppercase) language feature for the IBM MQ operations and control panels.
thlqual.SCSQPROC	Contains sample JCL and default system initialization data sets.
thlqual.SCSQSNLC	Contains the load modules for the Simplified Chinese versions of the IBM MQ modules that are required for special purpose function (for example the early code).
thlqual.SCSQSNLE	Contains the load modules for the U.S. English (mixed case) versions of the IBM MQ modules that are required for special purpose function (for example the early code).
thlqual.SCSQSNLF	Contains the load modules for the French versions of the IBM MQ modules that are required for special purpose function (for example the early code).
thlqual.SCSQSNLK	Contains the load modules for the Japanese versions of the IBM MQ modules that are required for special purpose function (for example the early code).
thlqual.SCSQSNLU	Contains the load modules for the U.S. English (uppercase) versions of the IBM MQ modules that are required for special purpose function (for example the early code).

Table 136. IBM MQ libraries that exist after installation (continued)

Name	Description
thlqual.SCSQTBLC	Contains ISPF tables to be included in the ISPTLIB concatenation if you are using the Simplified Chinese language feature for the IBM MQ operations and control panels.
thlqual.SCSQTBLE	Contains ISPF tables to be included in the ISPTLIB concatenation if you are using the U.S. English (mixed case) language feature for the IBM MQ operations and control panels.
thlqual.SCSQTBLF	Contains ISPF tables to be included in the ISPTLIB concatenation if you are using the French language feature for the IBM MQ operations and control panels.
thlqual.SCSQTBLK	Contains ISPF tables to be included in the ISPTLIB concatenation if you are using the Japanese language feature for the IBM MQ operations and control panels.
thlqual.SCSQTBLU	Contains ISPF tables to be included in the ISPTLIB concatenation if you are using the U.S. English (uppercase) language feature for the IBM MQ operations and control panels.

Note: Do not modify or customize any of these libraries. If you want to make changes, copy the libraries and make your changes to the copies.

Related concepts:

"Setting up communications with other queue managers" on page 1531
This section describes the IBM MQ for z/OS preparations you need to make before you can start to use distributed queuing.

"Using IBM MQ with IMS" on page 1564
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 1573
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 1575
Use this topic if you want to use IMS Open Transaction Manager Access exits with IBM MQ for z/OS.

Related reference:

"Upgrading and applying service to Language Environment or z/OS Callable Services" on page 1573
The actions you must take vary according to whether you use CALLLIBS or LINK, and your version of SMP/E.

Related information:

IBM MQ for z/OS concepts

Administering IBM MQ for z/OS

Customizing IBM MQ for z/OS

z/OS

Use this topic as a step by step guide for customizing your IBM MQ system.

This topic leads you through the various stages of customizing IBM MQ after you have successfully installed it. The installation process is described in the Program Directory, available to download from the IBM Publications Center.

Samples are supplied with IBM MQ to help you with your customization. The sample data set members have names beginning with the four characters CSQ4 and are in the library thlqual.SCSQPROC.

Before you perform the customization tasks described in this topic, there are a number of configuration options that you must consider because they affect the performance and resource requirements of IBM MQ for z/OS. For example, you must decide which globalization libraries you want to use.

V 9.0.0 If you want to automate some of the customization steps, see "Using IBM z/OSMF to automate IBM MQ" on page 1579.

Configuration options

For more information about these options, see Planning on z/OS.

The description of each task in this section indicates whether:

- The task is part of the process of customizing IBM MQ. That is, you perform the task once when you customize IBM MQ on the z/OS system. (In a parallel sysplex, you must perform the task for each z/OS system in the sysplex, and ensure that each z/OS system is set up identically.)
- The task is part of adding a queue manager. That is, you perform the task once for each queue manager when you add that queue manager.
- You need to perform the task when migrating. If you are migrating from a previous version of IBM MQ for z/OS, you might not need to perform all these tasks.

Review the tasks when you apply corrective maintenance to IBM MQ and when you install a new version or release of IBM MQ.

None of the tasks require you to perform an IPL of your z/OS system, if you use commands to change the various z/OS system parameters, and perform "Task 12: Update SYS1.PARMLIB members" on page 1479 as suggested.

To simplify operations and to aid with problem determination, ensure that all z/OS systems in a sysplex are set up identically, so that queue managers can be quickly created on any system in an emergency.

For ease of maintenance, consider defining aliases to refer to your IBM MQ libraries; for more information, see Using an alias to refer to an IBM MQ library.

Identify the national language support libraries

You need to specify the appropriate globalization libraries in the JCL that you want to use with IBM MQ (as described in the following sections). Each language is identified by a language letter:

C	Simplified Chinese
E	U.S. English (mixed case)
F	French
K	Japanese
U	U.S. English (uppercase)

Table 137. National language feature libraries

Description	Japanese	Simplified Chinese	U.S. English (mixed case)	U.S. English (uppercase)	French
Load modules	thlqual.SCSQANLK	thlqual.SCSQANLC	thlqual.SCSQANLE	thlqual.SCSQANLU	thlqual.SCSQANLF
ISPF messages	thlqual.SCSQMSGK	thlqual.SCSQMSGC	thlqual.SCSQMSGE	thlqual.SCSQMSGU	thlqual.SCSQMSGF
ISPF panels	thlqual.SCSQPNLK	thlqual.SCSQPNLC	thlqual.SCSQPNLE	thlqual.SCSQPNLU	thlqual.SCSQPNLF
Special purpose function (for example, early code)	thlqual.SCSQSNLK	thlqual.SCSQSNLC	thlqual.SCSQSNLE	thlqual.SCSQSNLU	thlqual.SCSQSNLF
ISPF tables	thlqual.SCSQTBLK	thlqual.SCSQTBLC	thlqual.SCSQTBLE	thlqual.SCSQTBLU	thlqual.SCSQTBLF

Customization summary

The following table lists all the steps required to customize IBM MQ for z/OS. It also indicates the following:

- Whether the step has to be performed once only, or repeated for each queue manager.
- Whether you need to repeat the step for each queue-sharing group, or omit the step if you are not using queue-sharing groups.
- Whether the step is required if you are migrating from a previous version of IBM MQ. Some steps might be needed, depending on what you decide about data set and queue manager names; these steps are marked 'Review'.

Table 138. Customization summary

Task	Required when migrating	Repeat for each queue manager	Queue-sharing groups
"Task 1: Identify the z/OS system parameters" on page 1463	Review	-	-
"Task 2: APF authorize the IBM MQ load libraries" on page 1464	Review	-	-
"Task 3: Update the z/OS link list and LPA" on page 1465	Review	-	-
"Task 4: Update the z/OS program properties table" on page 1467	-	-	-
"Task 5: Define the IBM MQ subsystem to z/OS" on page 1467	-	X	-
"Task 6: Create procedures for the IBM MQ queue manager" on page 1472	Review	X	-
"Task 7: Create procedures for the channel initiator" on page 1473	Review	X	-
"Task 8: Define the IBM MQ subsystem to a z/OS WLM service class" on page 1474	-	X	-
"Task 9: Set up the Db2 environment" on page 1475	Review	-	Omit if not used
"Task 10: Set up the coupling facility" on page 1476	Review	-	Repeat for each
"Task 11: Implement your ESM security controls" on page 1478	Review	X	X

Table 138. Customization summary (continued)

Task	**Required when migrating**	**Repeat for each queue manager**	**Queue-sharing groups**
"Task 12: Update SYS1.PARMLIB members" on page 1479	Review	-	-
"Task 13: Customize the initialization input data sets" on page 1479	X	X	-
"Task 14: Create the bootstrap and log data sets" on page 1482	-	X	-
"Task 15: Define your page sets" on page 1484	-	X	-
"Task 16: Add the IBM MQ entries to the Db2 tables" on page 1485	Review	X	Repeat for each
"Task 17: Tailor your system parameter module" on page 1485	X	X	-
"Task 18: Tailor the channel initiator parameters" on page 1508	X	X	-
"Task 19: Set up Batch, TSO, and RRS adapters" on page 1510	Review	-	-
"Task 20: Set up the operations and control panels" on page 1511	Review	-	-
"Task 21: Include the IBM MQ dump formatting member" on page 1512	X	-	-
"Task 22: Suppress information messages" on page 1513	-	-	-
"Task 23: Create procedures for Advanced Message Security" on page 1514	Review	X	-
"Task 24: Set up the started task user Advanced Message Security" on page 1514	Review	X	-
"Task 25: Grant RACDCERT permissions to the security administrator for Advanced Message Security" on page 1517	-	-	-
"Task 26: Grant users resource permissions for Advanced Message Security" on page 1517	-	-	-
V 9.0.1 "Create the Liberty server definition" on page 1518	X	-	-
V 9.0.1 "Create a procedure for the Liberty server" on page 1519	Review	-	-
V 9.0.1 Configuring the IBM MQ Console and REST API[1]	X	-	-
V 9.0.3 MQ Adv. VUE Configuring for use with the IBM Cloud Product Insights service	X	-	-

Note:

1. Task 30: Configuring the IBM MQ Console and REST API applies to distributed platforms, as well as z/OS.

Related concepts:

"Setting up communications with other queue managers" on page 1531
This section describes the IBM MQ for z/OS preparations you need to make before you can start to use distributed queuing.

"Using IBM MQ with IMS" on page 1564
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 1573
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 1575
Use this topic if you want to use IMS Open Transaction Manager Access exits with IBM MQ for z/OS.

Related reference:

"Upgrading and applying service to Language Environment or z/OS Callable Services" on page 1573
The actions you must take vary according to whether you use CALLLIBS or LINK, and your version of SMP/E.

Related information:

IBM MQ for z/OS concepts

Administering IBM MQ for z/OS

Program Directory for IBM MQ for z/OS

Task 1: Identify the z/OS system parameters

z/OS

Some of the tasks involve updating the z/OS system parameters. You need to know which ones were specified when the system IPL was performed.

- *You need to perform this task once for each z/OS system where you want to run IBM MQ.*
- *You might need to perform this task when migrating from a previous version.*

SYS1.PARMLIB(IEASYSpp) contains a list of parameters that point to other members of SYS1.PARMLIB (where pp represents the z/OS system parameter list that was used to perform an IPL of the system).

The entries you need to find are:

For "Task 2: APF authorize the IBM MQ load libraries" on page 1464:
PROG=xx or APF=aa point to the Authorized Program Facility (APF) authorized library list (member PROGxx or IEFAPFaa)

For "Task 3: Update the z/OS link list and LPA" on page 1465:
LNK=kk points to the link list (member LNKLSTkk) LPA=mm points to the LPA list (member LPALSTmm)

For "Task 4: Update the z/OS program properties table" on page 1467:
SCH=xx points to the Program Properties Table (PPT) (member SCHEDxx)

For "Task 5: Define the IBM MQ subsystem to z/OS" on page 1467:
SSN=ss points to the defined subsystem list (member IEFSSNss)

Task 2: APF authorize the IBM MQ load libraries

z/OS

APF-authorize various libraries. Some load modules might already be authorized.

- *You need to perform this task once for each z/OS system where you want to run IBM MQ.*
- *If you are using queue-sharing groups, you must ensure that the settings for IBM MQ are identical on each z/OS system in the sysplex.*
- *You might need to perform this task when migrating from a previous version.*

The IBM MQ load libraries thlqual.SCSQAUTH and thlqual.SCSQLINK must be APF-authorized. You must also APF-authorize the libraries for your national language feature (thlqual.SCSQANLx and thlqual.SCSQSNLx) and for the distributed queuing feature (thlqual.SCSQMVR1). If you are using Advanced Message Security you must also APF authorize the library thlqual.SDRQAUTH.

However, all load modules in the LPA are automatically APF-authorized. So are all members of the link list if the SYS1.PARMLIB member IEASYSpp contains the statement:

```
LNKAUTH=LNKLST
```

LNKAUTH=LNKLST is the default if LNKAUTH is not specified.

Depending on what you choose to put in the LPA or linklist (see “Task 3: Update the z/OS link list and LPA” on page 1465), you might not need to put the libraries in the APF link list

Note: You must APF-authorize all the libraries that you include in the IBM MQ STEPLIB. If you put a library that is not APF-authorized in the STEPLIB, the whole library concatenation loses its APF authorization.

The APF lists are in the SYS1.PARMLIB member PROGxx or IEAAPFaa. The lists contain the names of APF authorized z/OS libraries. The order of the entries in the lists is not significant. See the *MVS Initialization and Tuning Reference* manual for information about APF lists.

For more information about tuning your system, see SupportPac MP16

If you use PROGxx members with dynamic format, you need only issue the z/OS command `SETPROG APF,ADD,DSNAME=hlq.SCSQ` *`XXXX`*`,VOLUME=` *`YYYYYY`* for the changes to take effect: Where *XXXX* varies by the library name and where *YYYYY* is the volume. Otherwise, if you use static format or IEAAPFaa members, you must perform an IPL on your system.

Note that you must use the actual name of the library in the APF list. If you attempt to use the data set alias of the library, authorization fails.

Related concepts:
"Task 3: Update the z/OS link list and LPA"
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 1455
Use this topic when customizing your queue managers with details of installable features, national language features, and information about testing, and setting up security.

Task 3: Update the z/OS link list and LPA

z/OS

Update the LPA libraries with the new version of early code libraries. Other code can go in the link list or the LPA.

- You need to perform this task once for each z/OS system where you want to run IBM MQ.
- If you are using queue-sharing groups, you must ensure that the settings for IBM MQ are identical on each z/OS system in the sysplex.
- CD You might need to perform this task when migrating from a previous version. For details, see the *Program Directory for IBM MQ for z/OS Long Term Support Release V9.0.0* (GI13-3386-00), and *Program Directory for IBM MQ for z/OS Continuous Delivery V9.0.1* (GI13-3391-00), which can be downloaded from the IBM Publications Center

Note: The data set for LPA is version specific. If you are using an existing LPA in the system, contact your system administrator to decide which LPA to use.

Early code

Some IBM MQ load modules need to be added to MVS for IBM MQ to act as a subsystem. These modules are known as the Early code, and they can be executed even if a queue manager is not active. For example, when an operator command is issued on the console with an IBM MQ command prefix, this Early code will get control and check if it needs to start a queue manager, or to pass the request to a running queue manager. This code is loaded into the Link Pack Area (LPA). There is one set of Early modules, which are used for all queue managers, and these need to be at the highest level of IBM MQ. Early code from a higher version of IBM MQ will work with a queue manager with a lower version of IBM MQ, but not the opposite.

IBM MQ

The early code comprises the following load modules:
- CSQ3INI and CSQ3EPX in the library thqual.SCSQLINK
- CSQ3ECMX in the library thqual.SCSQSNL *x*, where *x* is your language letter.

CD IBM MQ includes a user modification that moves the contents of the thqual.SCSQSNL *i* library into the thqual.SCSQLINK and informs SMP/E. This user modification is called CSQ8UERL and is described in the *Program Directory for IBM MQ for z/OS Long Term Support Release V9.0.0* (GI13-3386), and *Program Directory for IBM MQ for z/OS Continuous Delivery V9.0.1* (GI13-3391-00), which can be downloaded from the IBM Publications Center.

When you have updated the early code in the LPA libraries, it is available from the next z/OS IPL (with the CLPA option) to all queue manager subsystems added during IPL from definitions in IEFSSNss members in SYS1.PARMLIB.

You can make it available immediately without an IPL for any new queue manager subsystem added later (as described in "Task 5: Define the IBM MQ subsystem to z/OS" on page 1467) by adding it to the LPA as follows:

- If you did not use CSQ8UERL, issue these z/OS commands:

  ```
  SETPROG LPA,ADD,MODNAME=(CSQ3INI,CSQ3EPX),DSNAME=thqual.SCSQLINK
  SETPROG LPA,ADD,MODNAME=(CSQ3ECMX),DSNAME=thqual.SCSQSNL x
  ```

- If you did use CSQ8UERL, you can load the early code into the LPA using the following z/OS command:

  ```
  SETPROG LPA,ADD,MASK=*,DSNAME=thqual.SCSQLINK
  ```

- If you are using Advanced Message Security you must also issue the following z/OS command to include an additional module in the LPA:

  ```
  SETPROG LPA,ADD,MODNAME=(CSQ0DRTM),DSNAME=thqual.SCSQLINK
  ```

If you have applied maintenance, or you intend to restart a queue manager with a later version or release of IBM MQ, the early code can be made available to queue manager subsystems that are already defined, provided the level of the early code when the z/OS system was started was at least that of Version 5.3. To make it available, use the following steps:

1. Add it to the LPA using z/OS SETPROG commands as described previously in this topic.
2. Stop the queue manager, using the IBM MQ command STOP QMGR.
3. Ensure that the qmgr.REFRESH.QMGR security profile is set up. See MQSC commands, profiles, and their access levels.
4. Refresh the early code for the queue manager using the IBM MQ command REFRESH QMGR TYPE(EARLY).
5. Restart the queue manager, using the IBM MQ command START QMGR.

The IBM MQ commands STOP QMGR, REFRESH QMGR, and START QMGR are described in MQSC commands.

Other code

All the IBM MQ supplied load modules in the following libraries are reentrant and can be placed in the LPA:

- SCSQAUTH
- SCSQANL *x*, where *x* is your language letter
- SCSQMVR1

Important: However, if you place the libraries in the LPA, whenever you apply maintenance, you have to copy any changed modules manually into the LPA. Therefore, it is preferable to put the IBM MQ load libraries in the link list, which can be updated after maintenance by issuing the z/OS command REFRESH LLA.

This is particularly recommended for SCSQAUTH so that you do not have to include it in several STEPLIBs. Only one language library, SCSQANL *x* should be placed in the LPA or link list. The link list libraries are specified in an LNKLSTkk member of SYS1.PARMLIB.

The distributed queuing facility and CICS bridge (but not the queue manager itself) need access to the Language Environment (LE) runtime library SCEERUN. If you use either of these facilities, you need to include SCEERUN in the link list.

Related concepts:

"Task 4: Update the z/OS program properties table"
Some additional PPT entries are needed for the IBM MQ queue manager.

Task 4: Update the z/OS program properties table

z/OS

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 1.12 and later versions, CSQYASCP is already defined to the operating system with the attributes detailed and no longer needs to be included in a SCHEDxx member of PARMLIB.

The IBM MQ queue manager controls swapping itself. However, if you have a heavily-loaded IBM MQ network and response time is critical, it might be advantageous to make the IBM MQ channel initiator nonswappable, by adding the CSQXJST PPT entry, at the risk of affecting the performance of the rest of your z/OS system.

If you require Advanced Message Security, add the CSQ0DSRV PPT entry.

Issue the z/OS command SET SCH= for these changes to take effect.

Related concepts:

"Task 5: Define the IBM MQ subsystem to z/OS"
Update the subsystem name table and decide on a convention for command prefix strings.

Task 5: Define the IBM MQ subsystem to z/OS

z/OS

Update the subsystem name table and decide on a convention for command prefix strings.

Repeat this task for each IBM MQ queue manager. You do not need to perform this task when migrating from a previous version.

Related concepts:
"Task 6: Create procedures for the IBM MQ queue manager" on page 1472
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: z/OS

When defining the IBM MQ subsystem you must add an entry to the subsystem name table.

The subsystem name table of z/OS, which is taken initially from the SYS1.PARMLIB member IEFSSNss, contains the definitions of formally defined z/OS subsystems. To define each IBM MQ subsystem, you must add an entry to this table, either by changing the IEFSSNss member of SYS1.PARMLIB, or, preferably, by using the z/OS command SETSSI.

IBM MQ subsystem initialization supports parallel processing, so IBM MQ subsystem definition statements can be added both above and below the BEGINPARALLEL keyword in the IEFSSNss table available at z/OS V1.12 and later.

If you use the SETSSI command, the change takes effect immediately, and there is no need to perform an IPL of your system. Ensure you update SYS1.PARMLIB as well, as described in "Task 12: Update SYS1.PARMLIB members" on page 1479 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 1469 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 1471 for information about system scope).

Figure 176 shows several examples of IEFSSNss statements.

```
CSQ1,CSQ3INI,'CSQ3EPX,+mqs1cpf,S'
CSQ2,CSQ3INI,'CSQ3EPX,+mqs2cpf,S'
CSQ3,CSQ3INI,'CSQ3EPX,++,S'
```

Figure 176. 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 139 gives a number of examples showing the associations of subsystem names and command prefix strings (CPFs), as defined by the statements in Figure 176.

Table 139. 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): z/OS

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 141 on page 1470.
- 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 141 on page 1470) 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 140 on page 1470.

Table 140. Example of CPF subset and superset rules

Subsystem name	CPF defined	Commands routed to
MQA	!A	MQA
MQB	!B	MQB
MQC1	!C1	MQC1
MQC2	!C2	MQC2
MQB1	!B1	MQB

Commands intended for subsystem MQB1 (using CPF !B1) are routed to subsystem MQB because the CPF for this subsystem is !B, a subset of !B1. For example, if you entered the command:

```
!B1  START  QMGR
```

subsystem MQB receives the command:

```
1 START QMGR
```

(which, in this case, it cannot deal with).

You can see which prefixes exist by issuing the z/OS command DISPLAY OPDATA.

If you are running in a sysplex, z/OS diagnoses any conflicts of this type at the time of CPF registration (see "CPFs in a sysplex environment" on page 1471 for information about CPF registration).

Table 141 shows the characters that you can use when defining your CPF strings:

Table 141. Valid character set for CPF strings

Character set	Contents
Alphabetic	Uppercase A through Z, lowercase a through z
Numeric	0 through 9
National (see note)	@ $ # (Characters that can be represented as hexadecimal values)
Special	. ⌂ ( ) * & + - = ¢ < | ! ; % _ ? : >

Note:

The system recognizes the following hexadecimal representations of the national characters: @ as X'7C', $ as X'5B', and # as X'7B'. In countries other than the U.S., the U.S. national characters represented on terminal keyboards might generate a different hexadecimal representation and cause an error. For example, in some countries the $ character might generate an X'4A'.

The semicolon (;) is valid as a CPF but on most systems, this character is the command delimiter.

CPFs in a sysplex environment: z/OS

Use this topic to understand how to use CPFs within the scope of a sysplex.

If used in a sysplex environment, IBM MQ registers your CPFs to enable you to enter a command from any console in the sysplex and route that command to the appropriate system for execution. The command responses are returned to the originating console.

Defining the scope for sysplex operation

Scope is used to determine the type of CPF registration performed by the IBM MQ subsystem when you are running IBM MQ in a sysplex environment.

Possible values for scope are as follows:

M System scope.

The CPF is registered with z/OS at system IPL time by IBM MQ and remains registered for the entire time that the z/OS system is active.

IBM MQ commands must be entered at a console connected to the z/OS image running the target subsystem, or you must use ROUTE commands to direct the command to that image.

Use this option if you are not running in a sysplex.

S Sysplex started scope.

The CPF is registered with z/OS when the IBM MQ subsystem is started, and remains active until the IBM MQ subsystem terminates.

You must use ROUTE commands to direct the original START QMGR command to the target system, but all further IBM MQ commands can be entered at any console connected to the sysplex, and are routed to the target system automatically.

After IBM MQ termination, you must use the ROUTE commands to direct subsequent START commands to the target IBM MQ subsystem.

X Sysplex IPL scope.

The CPF is registered with z/OS at system IPL time by IBM MQ and remains registered for the entire time that the z/OS system is active.

IBM MQ commands can be entered at any console connected to the sysplex, and are routed to the image that is executing the target system automatically.

An IBM MQ subsystem with a CPF with scope of S can be defined on one or more z/OS images within a sysplex, so these images can share a single subsystem name table. However, you must ensure that the initial START command is issued on (or routed to) the z/OS image on which you want the IBM MQ subsystem to run. If you use this option, you can stop the IBM MQ subsystem and restart it on a different z/OS image within the sysplex without having to change the subsystem name table or perform an IPL of a z/OS system.

An IBM MQ subsystem with a CPF with scope of X can only be defined on one z/OS image within a sysplex. If you use this option, you must define a unique subsystem name table for each z/OS image requiring IBM MQ subsystems with CPFs of scope X.

If you want to use the z/OS automatic restart manager (ARM) to restart queue managers in different z/OS images automatically, every queue manager must be defined in each z/OS image on which that queue manager might be restarted. Every queue manager must be defined with a sysplex-wide, unique 4-character subsystem name with a CPF scope of S.

Task 6: Create procedures for the IBM MQ queue manager

z/OS

Each IBM MQ subsystem needs a cataloged procedure to start the queue manager. You can create your own or use the IBM-supplied procedure library.

- *Repeat this task for each IBM MQ queue manager.*
- *You might need to modify the catalogued procedure when migrating from a previous version.*

For each IBM MQ subsystem defined in the subsystem name table, create a cataloged procedure in a procedure library for starting the queue manager. The IBM-supplied procedure library is called SYS1.PROCLIB, but your installation might use its own naming convention.

The name of the queue manager started task procedure is formed by concatenating the subsystem name with the characters MSTR. For example, subsystem CSQ1 has the procedure name CSQ1MSTR. You need one procedure for each subsystem you define.

Many examples and instructions in this product documentation assume that you have a subsystem called CSQ1 You might find these examples easier to use if a subsystem called CSQ1 is created initially for installation verification and testing purposes.

Two sample started task procedures are provided in thlqual.SCSQPROC. Member CSQ4MSTR uses one page set for each class of message, member CSQ4MSRR uses multiple page sets for the major classes of message. Copy one of these procedures to member xxxxMSTR (where xxxx is the name of your IBM MQ subsystem) of your SYS1.PROCLIB or, if you are not using SYS1.PROCLIB, your procedure library. Copy the sample procedure to a member in your procedure library for each IBM MQ subsystem that you define.

When you have copied the members, you can tailor them to the requirements of each subsystem, using the instructions in the member. For information about specifying region sizes below the 16 MB line, above the 16 MB line, and above the 2 GB bar, see Suggested region sizes. You can also use symbolic parameters in the JCL to allow the procedure to be modified when it is started. If you have several IBM MQ subsystems, you might find it advantageous to use JCL include groups for the common parts of the procedure, to simplify future maintenance.

If you are using queue-sharing groups, the STEPLIB concatenation must include the Db2 runtime target library SDSNLOAD, and it must be APF-authorized. This library is only required in the STEPLIB concatenation if it is not accessible through the link list or LPA.

If you are using Advanced Message Security the STEPLIB concatenation must include *thlqual*.SDRQAUTH and it must be APF authorized.

You can add the exit library (CSQXLIB) to this procedure later if you want to use queue manager exits. You need access to the Language Environment (LE) runtime library SCEERUN to do this; if it is not in your link list (SYS1.PARMLIB(LNKLSTkk)), concatenate it in the STEPLIB DD statement. You also must stop and restart your queue manager.

z/OS V 9.0.3 MQ Adv. VUE If you are running IBM MQ Advanced for z/OS, Value Unit Edition Version 9.0.3, and want to publish registration and usage data for your queue manager to the IBM Cloud Product Insights service on IBM Cloud (formerly Bluemix), you need to:

1. Uncomment the CSQ4INSC CSQINP2 concatenation DD card.

Sample `CSQ4INSC.jcl`, shipped in SCSQPROC, contains the definition for the SYSTEM.BLUEMIX.REGISTRATION.QUEUE, that is used for exchanging registration and usage data messages from the queue manager to the channel initiator.

Attention: By default, the queue is defined with DEFPSIST(NO) and messages are put to the queue with persistence as queue definition. This means that messages put to the queue will be non-persistent. However, you can customize the attributes specified in `CSQ4INSC.jcl`, as required, to achieve different behavior.

2. Define a QM INI data set to configure the IBM Cloud (formerly Bluemix) registration stanza; see Configuring the IBM Cloud (formerly Bluemix) registration stanza, and uncomment the CSQMQMIN DD card.

Note: You can make a note of the names of your bootstrap data set (BSDS), logs, and page sets for use in JCL and then define these sets at a later step in the process.

Related concepts:

"Task 7: Create procedures for the channel initiator"
For each IBM MQ subsystem, tailor a copy of CSQ4CHIN. Depending on what other products you are using, you might need to allow access to other data sets.

Task 7: Create procedures for the channel initiator

z/OS

For each IBM MQ subsystem, tailor a copy of CSQ4CHIN. Depending on what other products you are using, you might need to allow access to other data sets.

- *Repeat this task for each IBM MQ queue manager.*
- *You might need to perform this task when migrating from a previous version.*

You need to create a channel-initiator started task procedure for each IBM MQ subsystem that is going to use distributed queuing.

To do this:

1. Copy the sample started task procedure thlqual.SCSQPROC(CSQ4CHIN) to your procedure library. Name the procedure *xxxx* CHIN, where *xxxx* is the name of your IBM MQ subsystem (for example, CSQ1CHIN would be the channel initiator started task procedure for queue manager CSQ1).
2. Make a copy for each IBM MQ subsystem that you are going to use.
3. Tailor the procedures to your requirements using the instructions in the sample procedure CSQ4CHIN. You can also use symbolic parameters in the JCL to allow the procedure to be modified when it is started. This is described with the start options in the ../com.ibm.mq.adm.doc/q022070_.dita.

   Concatenate the distributed queuing library thlqual.SCSQMVR1.

   Access to the LE runtime library SCEERUN is required; if it is not in your link list (SYS1.PARMLIB(LNKLSTkk)), concatenate it in the STEPLIB DD statement.
4. Authorize the procedures to run under your external security manager.

The channel initiator is a long running address space. To prevent its termination after a restricted amount of CPU has been consumed, confirm that either:

- The default for started tasks in your z/OS system is unlimited CPU; a JES2 configuration statement for JOBCLASS(STC) with TIME=(1440,00) achieves this, or
- Explicitly add a TIME=1440, or TIME=NOLIMIT, parameter to the EXEC statement for CSQXJST.

You can add the exit library (CSQXLIB) to this procedure later if you want to use channel exits. You need to stop and restart your channel initiator to do this.

If you are using 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
- HFS file, /etc/resolv.conf
- //SYSTCPD DD statement
- //SYSTCPDD DD statement
- *jobname/userid*.TCPIP.DATA
- SYS1.TCPPARMS(TCPDATA)
- *zapname*.TCPIP.DATA

Some of these affect your started-task procedure JCL. For more information, see z/OS Communications Server: IP Configuration Guide.

Related concepts:

"Task 8: Define the IBM MQ subsystem to a z/OS WLM service class"
To give IBM MQ appropriate performance priority in the z/OS system, you must assign the queue manager and channel initiator address spaces to an appropriate z/OS workload management (WLM) service class. If you do not do this explicitly, inappropriate defaults might apply.

Task 8: Define the IBM MQ subsystem to a z/OS WLM service class

z/OS

To give IBM MQ appropriate performance priority in the z/OS system, you must assign the queue manager and channel initiator address spaces to an appropriate z/OS workload management (WLM) service class. If you do not do this explicitly, inappropriate defaults might apply.
- *Repeat this task for each IBM MQ queue manager.*
- *You do not need to perform this task when migrating from a previous version.*

Use the ISPF dialog supplied with WLM to perform the following tasks:
- Extract the z/OS WLM policy definition from the WLM couple data set.
- Update this policy definition by adding queue manager and channel initiator started task procedure names to the chosen service class
- Install the changed policy on the WLM couple data set

Then activate this policy using the z/OS command

```
V WLM,POLICY=policyname,REFRESH
```

See for more information on setting performance options.

Related concepts:
"Task 9: Set up the Db2 environment"
If you are using queue-sharing groups you must create the required Db2 objects by customizing and running a number of sample jobs.

Task 9: Set up the Db2 environment

z/OS

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.

V 9.0.4 IBM MQ provides two equivalent sets of jobs. Those with the CSQ45 prefix are for compatibility with earlier versions of IBM MQ and for use with Db2 version 11 and earlier. If you are setting up a new data-sharing group with Db2 V12 or later, you are encouraged to use the jobs with CSQ4X prefix, as these jobs exploit more recent Db2 capabilities for dynamic sizing and Universal Table Spaces.

You must establish an environment in which IBM MQ can access and execute the Db2 plans that are used for queue-sharing groups.

The following steps must be performed for each new Db2 data-sharing group. All the sample JCL is in thlqual.SCSQPROC.

1. Customize and execute sample JCL CSQ45CSG V 9.0.4 (or CSQ4XCSG) to create the storage group that is to be used for the IBM MQ database, table spaces, and tables.
2. Customize and execute sample JCL CSQ45CDB V 9.0.4 (or CSQ4XCDB) to create the database to be used by all queue managers that are connecting to this Db2 data-sharing group.
3. Customize and execute sample JCL CSQ45CTS V 9.0.4 (or CSQ4XCTS) to create the table spaces that contain the queue manager and channel initiator tables used for queue-sharing groups (to be created in step 1).
4. Customize and execute sample JCL CSQ45CTB V 9.0.4 (or CSQ4XCTB) to create the 12 Db2 tables and associated indexes. Do not change any of the row names or attributes.
5. Customize and execute sample JCL CSQ45BPL 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 for the:

   - LTS Long Term Support version in the LTS column.
   - CD Continuous Delivery version in the CD column, where `n` represents the CD release.

     At each release, `n` increments by one. For example, at IBM MQ Version 9.0.3, `CSQ5A90n` is `CSQ5A903`.

User	Plans (LTS)	Plans (CD)
Queue manager	CSQ5A 900, CSQ5C 900, CSQ5D 900, CSQ5K 900, CSQ5L 900, CSQ5M 900, CSQ5P 900, CSQ5R 900, CSQ5S 900, CSQ5T 900, CSQ5U 900, CSQ5W 900	CSQ5A 90n, CSQ5C 90n, CSQ5D 90n, CSQ5K 90n, CSQ5L 90n, CSQ5M 90n, CSQ5P 90n, CSQ5R 90n, CSQ5S 90n, CSQ5T 90n, CSQ5U 90n, CSQ5W 90n
SDEFS function of the CSQUTIL batch utility	CSQ52 900	CSQ52 90n
CSQ5PQSG and CSQJUCNV batch utilities	CSQ5B 900	CSQ5B 90n
CSQUZAP service utility	CSQ5Z 900	CSQ5Z 90n

In the event of a failure during Db2 setup, the following jobs can be customized and executed:

- CSQ45DTB to drop the tables and indexes.
- CSQ45DTS V 9.0.4 (or CSQ4XDTS) to drop the table spaces.
- CSQ45DDB V 9.0.4 (or CSQ4XDDB) to drop the database.
- CSQ45DSG V 9.0.4 (or CSQ4XDSG) 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 11.0.0* for more information about setting up Db2.

V 9.0.4 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:

"Task 10: Set up the coupling facility"
If you are using queue-sharing groups, define the coupling facility structures used by the queue managers in the queue-sharing group (QSG) in the coupling facility Resource Management (CFRM) policy data set, using IXCMIAPU.

Task 10: Set up the coupling facility

z/OS

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.

- *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.

Set up the SMDS environment

If you choose SMDS as the offload storage environment, you need to perform this process for each structure in the QSG.

Attention: If you are choosing the SMDS offload storage environment, you are still required to set up the Db2 environment for shared queues.

- Estimate structure and data set space requirements. See Shared message data set capacity considerations.
- Allocate and preformat data sets. See Creating a shared message data set.
- Use the following MQSC command to display the **CFLEVEL** and **OFFLOAD** status.

  ```
  DISPLAY CFSTRUCT(*) CFLEVEL OFFLOAD
  ```

  For more information about the **DISPLAY CFSTRUCT** command, see DISPLAY CFSTRUCT.
- Ensure that the coupling facility structure is defined with **CFLEVEL(5)** and **OFFLOAD(SMDS)** by using the following MQSC commands:

  ```
  ALTER CFSTRUCT(APP1) CFLEVEL(5)
  ```

  ```
  ALTER CFSTRUCT(APP1) OFFLOAD(SMDS)
  ```

  For more information about the **ALTER CFSTRUCT** command, see ALTER CFSTRUCT.

All the structures for the queue-sharing group start with the name of the queue-sharing group. Define the following structures:

- An administrative structure called *qsg-name* CSQ_ADMIN. This structure is used by IBM MQ itself and does not contain any user data.
- A system application structure called *qsg-name* CSQSYSAPPL. This structure is used by IBM MQ system queues to store state information.
- One or more structures used to hold messages for shared queues. These can have any name you choose up to 16 characters long.
  - The first four characters must be the queue-sharing group name. (If the queue-sharing group name is less than four characters long, it must be padded to four characters with @ symbols.)
  - The fifth character must be alphabetic and subsequent characters can be alphabetic or numeric. This part of the name (without the queue-sharing group name) is what you specify for the CFSTRUCT name when you define a shared queue, or a CF structure object.

  You can use only alphabetic and numeric characters in the names of structures used to hold messages for shared queues, you cannot use any other characters (for example, the _ character, which is used in the name of the administrative structure).

Sample control statements for IXCMIAPU are in data set thlqual.SCSQPROC(CSQ4CFRM). Customize these and add them to your IXCMIAPU job for the coupling facility and run it.

When you have defined your structures successfully, activate the CFRM policy that is being used. To do this, issue the following z/OS command:

```
SETXCF START,POLICY,TYPE=CFRM,POLNAME= policy-name
```

See the Defining coupling facility resources for information about planning CF structures and their sizes.

Related concepts:

"Task 11: Implement your ESM security controls"
Implement security controls for queue-sharing groups, the channel initiator, and all queue managers accessing the coupling facility list structures.

Task 11: Implement your ESM security controls

z/OS

Implement security controls for queue-sharing groups, the channel initiator, and all queue managers accessing the coupling facility list structures.

- *Repeat this task for each IBM MQ queue manager or queue-sharing group.*
- *You might need to perform this task when migrating from a previous version.*

If you use RACF as your external security manager, see Setting up security on z/OS, which describes how to implement these security controls.

If you are using queue-sharing groups, ensure that the user IDs associated with the queue manager, channel initiator, and the utilities (as specified in task 9, step 6 on page 1475) have authority to establish an RRSAF connection to each Db2 subsystem with which you want to establish a connection. The RACF profile to which the user ID requires READ access is *Db2ssid*.RRSAF in the DSNR resource class.

If you are using the channel initiator, you must also do the following:

- If your subsystem has connection security active, define a connection security profile ssid.CHIN to your external security manager (see Connection security profiles for the channel initiator for information about this).
- If you are using 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 UNIX System Services, as described in the *OS/390® 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.

Those queue managers that will access the coupling facility list structures require the appropriate security access. The RACF class is FACILITY. The queue manager user ID requires ALTER access to the IXLSTR.`structure-name` profile.

Before you start the queue manager, set up IBM MQ data set and system security by:

- Authorizing the queue manager started task procedure to run under your external security manager.
- Authorizing access to the queue manager data sets.

For details about how to do this, see Security installation tasks for z/OS(r).

If you are using RACF, provided you use the RACF STARTED class, you do not need to perform an IPL of your system (see RACF authorization of started-task procedures).

Related concepts:

"Task 12: Update SYS1.PARMLIB members"
To ensure that your changes remain in effect after an IPL, you must update some members of SYS1.PARMLIB

Task 12: Update SYS1.PARMLIB members

z/OS

To ensure that your changes remain in effect after an IPL, you must update some members of SYS1.PARMLIB

- *You need to perform this task once for each z/OS system where you want to run IBM MQ.*
- *If you are using queue-sharing groups, you must ensure that the settings for IBM MQ are identical on each z/OS system in the sysplex.*
- *You might need to perform this task when migrating from a previous version.*

Update SYS1.PARMLIB members as follows:

1. Update member IEFSSNss as described in "Task 5: Define the IBM MQ subsystem to z/OS" on page 1467.
2. Change IEASYSpp so that the following members are used when an IPL is performed:
   - the PROGxx or IEAAPFaa members used in "Task 2: APF authorize the IBM MQ load libraries" on page 1464
   - the LNKLSTkk and LPALSTmm members used in "Task 3: Update the z/OS link list and LPA" on page 1465
   - the SCHEDxx member used in "Task 4: Update the z/OS program properties table" on page 1467
   - the IEFSSNss member used in "Task 5: Define the IBM MQ subsystem to z/OS" on page 1467

Related concepts:

"Task 13: Customize the initialization input data sets"
Make working copies of the sample initialization input data sets and tailor them to suit your system requirements.

Task 13: Customize the initialization input data sets

z/OS

Make working copies of the sample initialization input data sets and tailor them to suit your system requirements.

- *Repeat this task for each IBM MQ queue manager.*
- *You need to perform this task when migrating from a previous version.*

Each IBM MQ queue manager gets its initial definitions from a series of commands contained in the IBM MQ *initialization input data sets*. These data sets are referenced by the DDnames CSQINP1, CSQINP2 and CSQINPT defined in the queue manager started task procedure.

Responses to these commands are written to the initialization output data sets referenced by the DDnames CSQOUT1, CSQOUT2 and CSQOUTT.

To preserve the originals, make working copies of each sample. Then you can tailor the commands in these working copies to suit your system requirements.

If you use more than one IBM MQ subsystem, if you include the subsystem name in the high-level qualifier of the initialization input data set name, you can identify the IBM MQ subsystem associated with each data set more easily.

Refer to the following topics for further information about the samples:
- Initialization data set formats
- Using the CSQINP1 sample
- Using the CSQINP2 samples
- Using the CSQINPX sample
- Using the CSQINPT sample

Initialization data set formats

The initialization input data sets can be partitioned data set (PDS) members or sequential data sets. They can be a concatenated series of data sets. Define them with a record length of 80 bytes, where:
- Only columns 1 through 72 are significant. Columns 73 through 80 are ignored.
- Records with an asterisk (*) in column 1 are interpreted as comments and are ignored.
- Blank records are ignored.
- Each command must start on a new record.
- A trailing - means continue from column 1 of the next record.
- A trailing + means continue from the first non-blank column of the next record.
- The maximum number of characters permitted in a command is 32 762.

The initialization output data sets are sequential data sets, with a record length of 125, a record format of VBA, and a block size of 629.

Using the CSQINP1 sample

Data set thlqual.SCSQPROC holds two members which contain definitions of buffer pools, page set to buffer pool associations, and an ALTER SECURITY command.

Member CSQ4INP1 uses one page set for each class of message. Member CSQ4INPR uses multiple page sets for the major classes of message.

Include the appropriate sample in the CSQINP1 concatenation of your queue manager started task procedure.

Notes:
1. IBM MQ supports up to 16 buffer pools (zero through 15). If OPMODE is set to OPMODE=(NEWFUNC, 800), 100 buffer pools are supported in the range zero through 99. The DEFINE BUFFPOOL command can only be issued from a CSQINP1 initialization data set. The definitions in the sample specify four buffer pools.
2. Each page set used by the queue manager must be defined in the CSQINP1 initialization data set by using the DEFINE PSID command. The page set definition associates a buffer pool ID with a page set. If no buffer pool is specified, buffer pool zero is used by default.

   Page set zero (00) must be defined. It contains all the object definitions. You can define up to 100 page sets for each queue manager.
3. The ALTER SECURITY command can be used to alter the security attributes TIMEOUT and INTERVAL. In CSQ4INP1, the default values are defined as 54 for TIMEOUT and 12 for INTERVAL.

See the Planning on z/OS for information about organizing buffer pools and page sets.

If you change the buffer pool and page set definitions dynamically while the queue manager is running, you should also update the CSQINP1 definitions. The changes are only retained for a cold start of IBM MQ, unless the buffer pool definition includes the REPLACE attribute.

Using the CSQINP2 samples

This table lists the members of thlqual.SCSQPROC that can be included in the CSQINP2 concatenation of your queue manager started task procedure, with a description of their function. The naming convention is CSQ4INS*. CSQ4INY* will need to be modified for YOUR configuration. You should avoid changing CSQINS* members because you will need to reapply any changes when you migrate to the next release. Instead, you can put DEFINE or ALTER commands in CSQ4INY* members.

Table 142. Members of thlqual.SCSQPROC

Member name	Description
CSQ4INSG	System object definitions.
CSQ4INSA	System object and default rules for channel authentication.
CSQ4INSX	System object definitions.
CSQ4INSS	Customize and include this member if you are using queue-sharing groups.
CSQ4INSJ	Customize and include this member if you are using publish/subscribe using JMS.
CSQ4INSM	System object definitions for advanced message security.
CSQ4INSR	Customize and include this member if you are using WebSphere Application Server, or the queued publish/subscribe interface supported by the queued publish/subscribe daemon in IBM MQ V7 or later.
CSQ4DISP	CSQINP2 sample for displaying object definitions.
CSQ4INYC	Clustering definitions.
CSQ4INYD	Distributed queuing definitions.
CSQ4INYG	General definitions.
CSQ4INYR	Storage class definitions, using multiple page sets for the major classes of message.
CSQ4INYS	Storage class definitions, using one page set for each class of message.

You need to define objects once only, not each time that you start a queue manager, so it is not necessary to include these definitions in CSQINP2 every time. If you do include them every time, you are attempting to define objects that already exist, and you will get messages similar to the following:

```
CSQM095I +CSQ1 CSQMAQLC QLOCAL(SYSTEM.DEFAULT.LOCAL.QUEUE) ALREADY EXISTS
CSQM090E +CSQ1 CSQMAQLC FAILURE REASON CODE X'00D44003'
CSQ9023E +CSQ1 CSQMAQLC ' DEFINE QLOCAL' ABNORMAL COMPLETION
```

The objects are not damaged by this failure. If you want to leave the SYSTEM definitions data set in the CSQINP2 concatenation, you can avoid the failure messages by specifying the REPLACE attribute against each object.

Using the CSQINPX sample

Sample thlqual.SCSQPROC(CSQ4INPX) contains a set of commands that you might want to execute each time the channel initiator starts. These are typically channel-related commands such as START LISTENER, which are required every time the channel initiator starts, rather than whenever the queue manager starts, and which are not allowed in the input data sets CSQINP1 or CSQINP2. You must customize this sample before use; you can then include it in the CSQINPX data set for the channel initiator.

The IBM MQ commands contained in the data set are executed at the end of channel initiator initialization, and output is written to the data set specified by the CSQOUTX DD statement. The output is like that produced by the COMMAND function of the IBM MQ utility program (CSQUTIL). See The CSQUTIL utility for more details.

You can specify any of the IBM MQ commands that can be issued from CSQUTIL, not only the channel commands. You can enter commands from other sources while CSQINPX is being processed. All commands are issued in sequence, regardless of the success of the previous command.

To specify a command response time, you can use the pseudo-command COMMAND as the first command in the data set. This takes a single optional keyword RESPTIME(*nnn*), where *nnn* is the time, in seconds, to wait for a response to each command. This is in the range 5 through 999; the default is 30.

If IBM MQ detects that the responses to four commands have taken too long, processing of CSQINPX is stopped and no further commands are issued. The channel initiator is not stopped, but message CSQU052E is written to the CSQOUTX data set, and message CSQU013E is sent to the console.

When IBM MQ has completed processing of CSQINPX successfully, message CSQU012I is sent to the console.

Using the CSQINPT sample

This table lists the members of thlqual.SCSQPROC that can be included in the CSQINPT concatenation of your queue manager started task procedure, with a description of their function.

Table 143. Members of thlqual.SCSQPROC

Member name	Description
CSQ4INST	System default subscription definition.
CSQ4INYT	Publish/Subscribe definitions.

The IBM MQ commands contained in the data set are executed when publish/subscribe initialization completes, and output is written to the data set specified by the CSQOUTT DD statement. The output is like that produced by the COMMAND function of the IBM MQ utility program (CSQUTIL). See The CSQUTIL utility for more details.

Related concepts:
"Task 14: Create the bootstrap and log data sets"
Use the supplied program CSQJU003 to prepare the bootstrap data sets (BSDSs) and log data sets.

Task 14: Create the bootstrap and log data sets

z/OS

Use the supplied program CSQJU003 to prepare the bootstrap data sets (BSDSs) and log data sets.

- *Repeat this task for each IBM MQ queue manager.*
- *You do not need to perform this task when migrating from a previous version.*

The sample JCL and Access Method Services (AMS) control statements to run CSQJU003 to create a single or dual logging environment are held in thlqual.SCSQPROC(CSQ4BSDS). Customize and run this job to create your BSDSs and logs and to preformat the logs.

Important: You should use the newest version of CSQ4BSDS, or update your JCL manually to use RECORDS(850 60).

The started task procedure, CSQ4MSTR, described in "Task 6: Create procedures for the IBM MQ queue manager" on page 1472, refers to BSDSs in statements of the form:

```
//BSDS1   DD DSN=++HLQ++.BSDS01,DISP=SHR
//BSDS2   DD DSN=++HLQ++.BSDS02,DISP=SHR
```

The log data sets are referred to by the BSDSs.

Note:

1. The BLKSIZE must be specified on the SYSPRINT DD statement in the LOGDEF step. The BLKSIZE must be 629.
2. To help identify bootstrap data sets and log data sets from different queue managers, include the subsystem name in the high level qualifier of these data sets.
3. If you are using queue-sharing groups, you must define the bootstrap and log data sets with SHAREOPTIONS(2 3).

See Planning on z/OS for information about planning bootstrap and log data sets and their sizes.

For IBM MQ Version 8.0, the 8 byte log RBA enhancement improves the availability of a queue manager, as described in Larger log Relative Byte Address. To enable 8 byte log RBA on a queue manager before the queue manager is first started, perform the following steps after creating your logging environment.

1. Using **`IDCAMS ALTER`**, rename the version 1 format BSDSs (created using the CSQJU003 program) to something like `++HLQ++.V1.BSDS01`.

   Note: Ensure that you rename the data and index components as well as the VSAM cluster.
2. Allocate new BSDSs with the same attributes as the ones already defined. These will become the version 2 format BSDSs that will be used by the queue manager when it is started.
3. Run the BSDS conversion utility (CSQJUCNV) to convert the version 1 format BSDSs to the new version 2 format BSDSs.
4. Once the conversion completes successfully, delete the version 1 format BSDSs.
5. In order to use 8 byte log RBA, ensure that Version 8.0 new functions are enabled with OPMODE as described in "Task 17: Tailor your system parameter module" on page 1485.

Note: CD If the queue manager is in a queue-sharing group, all queue managers in the queue-sharing group must have been started with OPMODE=(NEWFUNC,800) or OPMODE=(NEWFUNC,900) before 8 byte log RBA can be enabled.

Related concepts:

"Task 15: Define your page sets"
Define page sets for each queue manager using one of the supplied samples.

Task 15: Define your page sets

z/OS

Define page sets for each queue manager using one of the supplied samples.

- *Repeat this task for each IBM MQ queue manager.*
- *You do not need to perform this task when migrating from a previous version.*

Define separate page sets for each IBM MQ queue manager. thlqual.SCSQPROC(CSQ4PAGE) and thlqual.SCSQPROC(CSQ4PAGR) contain JCL and AMS control statements to define and format page sets. Member CSQ4PAGE uses one page set for each class of message, member CSQ4PAGR uses multiple page sets for the major classes of message. The JCL runs the supplied utility program CSQUTIL. Review the samples and customize them for the number of page sets you want and the sizes to use. See the Planning on z/OS for information about page sets and how to calculate suitable sizes.

The started task procedure CSQ4MSTR described in "Task 6: Create procedures for the IBM MQ queue manager" on page 1472 refers to the page sets, in a statement of the form:

```
//CSQP00 nn DD DISP=OLD,DSN= xxxxxxxxx
```

where *nn* is the page set number between 00 and 99, and *xxxxxxxxx* is the data set that you define.

Note:

1. If you intend to use the dynamic page set expansion feature, ensure that secondary extents are defined for each page set. thlqual.SCSQPROC(CSQ4PAGE) shows how to do this.
2. To help identify page sets from different queue managers, include the subsystem name in the high level qualifier of the data set associated with each page set.
3. If you intend to allow the FORCE option to be used with the FORMAT function of the utility program CSQUTIL, you must add the REUSE attribute on the AMS DEFINE CLUSTER statement. This is described in the Administering IBM MQ for z/OS.
4. If your page sets are to be larger than 4 GB you must use the Storage Management System (SMS) EXTENDED ADDRESSABILITY function.

Related concepts:

"Task 16: Add the IBM MQ entries to the Db2 tables"
If you are using queue-sharing groups, run the CSQ5PQSG utility to add queue-sharing group and queue manager entries to the IBM MQ tables in the Db2 data-sharing group.

Task 16: Add the IBM MQ entries to the Db2 tables

z/OS

If you are using queue-sharing groups, run the CSQ5PQSG utility to add queue-sharing group and queue manager entries to the IBM MQ tables in the Db2 data-sharing group.

- *Repeat this task for each IBM MQ queue-sharing group and each queue manager.*
- *You might need to perform this task when migrating from a previous version.*
- *Omit this task if you are not using queue-sharing groups.*

  If you later want to use queue-sharing groups, perform this task at that time.

Run CSQ5PQSG for each queue-sharing group and each queue manager that is to be a member of a queue-sharing group. (CSQ5PQSG is described in the Administering IBM MQ for z/OS.)

Perform the following actions in the specified order:

1. Add a queue-sharing group entry into the IBM MQ Db2 tables using the ADD QSG function of the CSQ5PQSG program. A sample is provided in thlqual.SCSQPROC(CSQ45AQS).

   Perform this function once for each queue-sharing group that is defined in the Db2 data-sharing group. The queue-sharing group entry must exist before adding any queue manager entries that reference the queue-sharing group.
2. Add a queue manager entry into the IBM MQ Db2 tables using the ADD QMGR function of the CSQ5PQSG program. A sample is provided in thlqual.SCSQPROC(CSQ45AQM).

   Perform this function for each queue manager that is to be a member of the queue-sharing group.

   Note:

   a. A queue manager can only be a member of one queue-sharing group.
   b. You must have RRS running to be able to use queue-sharing groups.

Related concepts:

"Task 17: Tailor your system parameter module"
The IBM MQ system parameter module controls the logging, archiving, tracing, and connection environments that IBM MQ uses in its operation. A default module is supplied, or you can create your own using supplied JCL and assembler source modules.

Task 17: Tailor your system parameter module

z/OS

The IBM MQ system parameter module controls the logging, archiving, tracing, and connection environments that IBM MQ uses in its operation. A default module is supplied, or you can create your own using supplied JCL and assembler source modules.

- *Repeat this task for each IBM MQ queue manager, as required.*
- *You need to perform this task when migrating from a previous version. For details, see Maintaining and migrating.*
- *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 1487. If you are configuring this queue manager for the first time, complete the whole of this task.*

The system parameter module has V 9.0.3 four macros as follows:

Macro name	Purpose
CSQ6SYSP	Specifies the connection and tracing parameters, see “Using CSQ6SYSP” on page 1487
CSQ6LOGP	Controls log initialization, see “Using CSQ6LOGP” on page 1497
CSQ6ARVP	Controls archive initialization, see “Using CSQ6ARVP” on page 1501
V 9.0.3 CSQ6USGP	Controls usage recording, see “Using CSQ6USGP” on page 1507

IBM MQ supplies a default system parameter module, CSQZPARM, which is invoked automatically if you issue the START QMGR command (without a PARM parameter) to start an instance of IBM MQ. CSQZPARM is in the APF-authorized library thlqual.SCSQAUTH also supplied with IBM MQ. The values of these parameters are displayed as a series of messages when you start IBM MQ.

See START QMGR for more information about how this command is used.

Creating your own system parameter module

If CSQZPARM does not contain the system parameters you want, you can create your own system parameter module using the sample JCL provided in thlqual.SCSQPROC(CSQ4ZPRM).

To create your own system parameter module:

1. Make a working copy of the JCL sample.
2. Edit the parameters for each macro in the copy as required. If you remove any parameters from the macro calls, the default values are automatically picked up at run time.
3. Replace the placeholder `++NAME++` with the name that the load module is to take (this can be CSQZPARM).
4. If your assembler is not high-level assembler, change the JCL as required by your assembler.
5. Run the JCL to assemble and link edit the tailored versions of the system parameter macros to produce a load module. This is the new system parameter module with the name that you have specified.
6. Put the load module produced in an APF-authorized user library.
7. Add user READ access to the APF-authorized user library.
8. Include this library in the IBM MQ queue manager started task procedure STEPLIB. This library name must come before the library thlqual.SCSQAUTH in STEPLIB.
9. Invoke the new system parameter module when you start the queue manager. For example, if the new module is named NEWMODS, issue the command:

```
START QMGR PARM(NEWMODS)
```

10. Ensure successful completion of the command by checking the job log. There should be an entry in the log similar to the following:
    ```
    CSQ9022I CDL1 CSQYASCP 'START QMGR' NORMAL COMPLETION
    ```

You can also specify the parameter module name in the queue manager startup JCL. For more information, see Starting and stopping a queue manager.

Note: If you choose to name your module CSQZPARM, you do not need to specify the PARM parameter on the START QMGR command.

Fine tuning a system parameter module

IBM MQ also supplies a set of three assembler source modules, which can be used to fine-tune an existing system parameter module. These modules are in library thlqual.SCSQASMS. Typically, you use these modules in a test environment to change the default parameters in the system parameter macros. Each source module calls a different system parameter macro:

This assembler source module...	Calls this macro...
CSQFSYSP	CSQ6SYSP (connection and tracing parameters)
CSQJLOGP	CSQ6LOGP (log initialization)
CSQJARVP	CSQ6ARVP (archive initialization)

This is how you use these modules:
1. Make working copies of each assembler source module in a user assembler library.
2. Edit your copies by adding or altering the values of any parameters as required.
3. Assemble your copies of any edited modules to create object modules in a user object library.
4. Link edit these object code modules with an existing system parameter module to produce a load module that is the new system parameter module.
5. Ensure that new system parameter module is a member of a user authorized library.
6. Include this library in the queue manager started task procedure STEPLIB. This library must come before the library thlqual.SCSQAUTH in STEPLIB.
7. Invoke the new system parameter module by issuing a START QMGR command, specifying the new module name in the PARM parameter, as before.

A sample usermod is provided in member CSQ4UZPR of SCSQPROC which demonstrates how to manage customized system parameters under SMP/E control.

Altering system parameters

You can alter some system parameters while a queue manager is running; see the SET SYSTEM, SET LOG, and SET ARCHIVE commands.

Put the SET commands in your initialization input data sets so that they take effect every time you start the queue manager.
Related concepts:
"Task 18: Tailor the channel initiator parameters" on page 1508
Use ALTER QMGR to customize the channel initiator to suit your requirements.

Using CSQ6SYSP: z/OS

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 144 on page 1488. If you want to change any of these values, see the detailed descriptions of the parameters.

Table 144. Default values of CSQ6SYSP parameters

Parameter	Description	Default value	SET command
ACELIM	Size of ACE storage pool in 1 KB blocks.	0 (no limit)	✓
CLCACHE	Specifies the type of cluster cache to use.	STATIC	-
CMDUSER	The default user ID for command security checks.	CSQOPR	-
CONNSWAP	Specifies whether jobs that are issuing certain IBM MQ API calls are swappable or non-swappable.	YES	-
EXCLMSG	Specifies a list of messages to be excluded from any log. Messages in this list are not sent to the z/OS console and hardcopy log. As a result using the EXCLMSG parameter to exclude messages is more efficient from a CPU perspective than using the methods described in "Task 22: Suppress information messages" on page 1513	( )	✓
EXITLIM	Time (in seconds) for which queue manager exits can run during each invocation.	30	-
EXITTCB	How many started server tasks to use to run queue manager exits.	8	-
LOGLOAD	Number of log records written by IBM MQ between the start of one checkpoint and the next.	500 000	✓
MULCCAPT	Determines the Measured Usage Pricing property which controls the algorithm for gathering data used by Measured Usage License Charging (MULC).	See parameter description	-
CD OPMODE	Controls the operation mode of the queue manager.	See parameter description	-
OTMACON	OTMA connection parameters.	See parameter description	-
QINDXBLD	Determines whether queue manager restart waits until all indexes are rebuilt, or completes before all indexes are rebuilt.	WAIT	-
QMCCSID	Coded character set identifier for the queue manager.	Zero	-
QSGDATA	Queue-sharing group parameters.	See parameter description	-
RESAUDIT	RESLEVEL auditing parameter.	YES	-
ROUTCDE	Message routing code assigned to messages not solicited from a specific console.	1	-
SERVICE	Reserved for use by IBM.	0	✓
SMFACCT	Specifies whether SMF accounting data is to be collected when the queue manager is started. Note that class 4 channel accounting data is collected only when the channel initiator is started.	NO	-
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	-

Table 144. Default values of CSQ6SYSP parameters (continued)

Parameter	Description	Default value	SET command
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	Default time, in minutes, between each gathering of statistics.	30	✓
TRACSTR	Specifies whether tracing is to be started automatically.	NO	-
TRACTBL	Size of trace table, in 4 KB blocks, to be used by the global trace facility.	99 (396 KB)	✓
WLMTIME	Time between scanning the queue index for WLM-managed queues.	30	-
WLMTIMU	Units (minutes or seconds) for WLMTIME.	MINS	-

ACELIM

Specifies the maximum size of the ACE storage pool in 1 KB blocks. The number must be in the range 0-999999. The default value of zero means no imposed constraint, beyond what is available in the system.

You should only set a value for ACELIM on queue managers that have been identified as using exorbitant quantities of ECSA storage. Limiting the ACE storage pool has the effect of limiting the number of connections in the system, and so, the amount of ECSA storage used by a queue manager.

Once the queue manager reaches the limit it is not possible for applications to obtain new connections. The lack of new connections causes failures in MQCONN processing, and applications coordinated through RRS are likely to experience failures in any IBM MQ API.

An ACE represents approximately 8% 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 64000K; that is 5120 multiplied by 12.5.

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 410 is required.

You can use SMF 115 subtype 5 records, produced by statistics CLASS(3) trace, to monitor the size of the 'ACE/PEB' storage pool, and hence set an appropriate value for ACELIM.

You can obtain the total amount of ECSA storage used by the queue manager, for control blocks, from SMF 115 subtype 7 records, written by statistics CLASS(2) trace; that is the first two elements in QSRSPHBT added together.

Note that, you should consider setting ACELIM as a mechanism to protect a z/OS image from a badly behaving queue manager, rather than as a means to control application connections to a queue manager.

CLCACHE

Specifies the type of cluster cache to use. See “Configuring a queue manager cluster” on page 1063 for more information.

STATIC

When the cluster cache is static, its size is fixed at queue manager start-up, enough for the current amount of cluster information plus some space for expansion. The size cannot increase while the queue manager is active. This is the default.

DYNAMIC
When the cluster cache is dynamic, the initial size allocated at queue manager startup can be increased automatically if required while the queue manager is active.

CMDUSER
Specifies the default user ID used for command security checks. This user ID must be defined to the ESM (for example, RACF). Specify a name of 1 through 8 alphanumeric characters. The first character must be alphabetic.

The default is CSQOPR.

CONNSWAP
Specifies whether batch jobs that are issuing certain IBM MQ API calls are swappable or non-swappable for the duration of the IBM MQ API request. Specify one of the following values:

NO Jobs are non-swappable during certain IBM MQ API calls.

YES Jobs are swappable during all IBM MQ API calls.

The default value is YES.

Use this parameter if low-priority jobs are swapped out while holding IBM MQ resources that other jobs or tasks might be waiting for. This parameter replaces the service parameter that was included IBM WebSphere MQ Version 7.0.1; the service parameter is no longer in use.

IBM MQ views WebSphere Application Server as part of an RRSBATCH environment. From IBM WebSphere MQ Version 7.1, when the CONNSWAP keyword is used, it is applied to any application in a BATCH or RRSBATCH environment. The CONNSWAP keyword is also applicable to TSO users, however, it is not applicable for CICS or IMS applications. CONNSWAP changes are implemented when a recycle of the queue manager takes place. A recycle is required after the keyword change is made, because the CSQ6SYSP macro is reassembled, and the queue manager restarted using the load module which is updated by the macro.

Alternatively, the WebSphere Application Server address space can be made non-swappable by using PPT.

EXCLMSG
Specifies a list of error messages to be excluded.

This list is dynamic and is updated using the SET SYSTEM command.

The default value is an empty list ( ).

Messages are supplied without the CSQ prefix and without the action code suffix (I-D-E-A). For example, to exclude message CSQX500I, add X500 to this list. This list can contain a maximum of 16 message identifiers.

To be eligible to be included in the list, the message must be issued after normal 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

OPMODE=(Mode,*VerificationLevel*)

OPMODE specifies the operation mode of the queue manager.

LTS The default setting of **OPMODE** for IBM MQ Version 9.0.0 is `OPMODE=(COMPAT,900)` .

▶ CD The only valid setting of **OPMODE**, for a Continuous Delivery (CD) release of IBM MQ, is `OPMODE=(NEWFUNC,90x)`. For example, at IBM MQ Version 9.0.1, you must specify `OPMODE=(NEWFUNC,901)`.

Mode
Specifies the requested operation mode. The values are as follows:

COMPAT
The queue manager runs in compatibility mode. Certain new functions are not available. The queue manager can be migrated back to an earlier release.

▶ LTS

Important: This value is valid only for a Long Term Support (LTS) release of IBM MQ.

NEWFUNC
All new functions provided in this level of code are available. The queue manager cannot be migrated back to an earlier release.

VerificationLevel

VerificationLevel is a `Version.Release.Modification` (VRM) code, without punctuation; `900`, for example.

The value of *VerificationLevel* ensures that the **CSQ6SYSP** parameters are coded for use with the level of **CSQ6SYSP** macro being compiled. If *VerificationLevel* does not match the VRM level of `SCSQMACS` used for **CSQ6SYSP**, then a compile-time error is reported. The *VerificationLevel* is compiled into the parameter module.

▶ LTS At queue manager start up, if the *VerificationLevel* does not match the release level of the queue manager, for an LTS release, then `COMPAT` mode is forced.

▶ CD For a queue manager started at a CD release, if *VerificationLevel* refers to a CD release, but does not match the release level of the queue manager, `OPMODE=(NEWFUNC,`*vrm*`)` takes effect, where *vrm* is the CD release level of the queue manager.

For example, a Version 9.0.2 queue manager started with `OPMODE=(NEWFUNC,901)`, behaves as if `OPMODE=(NEWFUNC,902)` had been specified.

If *VerificationLevel* refers to an LTS release, the queue manager will not start at a CD release.

The intent of the *VerificationLevel* parameter is to avoid inadvertent and irreversible setting of OPMODE to `NEWFUNC`. The mistake might occur when migrating to a newer version of IBM MQ using **CSQ6SYSP** statements prepared for an older version of the queue manager. It might also occur using a **CSQ6SYSP** parameter module built with an older version of the `SCSQMACS` macros.

OTMACON
OTMA parameters. This keyword takes five positional parameters::

OTMACON = (Group,Member,Druexit,Age,Tpipepfx)

Group This is the name of the XCF group to which this particular instance of IBM MQ belongs.

It can be 1 through 8 characters long and must be entered in uppercase characters.

The default is blanks, which indicates that IBM MQ must not attempt to join an XCF group.

Member
This is the member name of this particular instance of IBM MQ within the XCF group.

It can be 1 through 16 characters long and must be entered in uppercase characters.

The default is the 4-character queue manager name.

Druexit
This specifies the name of the OTMA destination resolution user exit to be run by IMS.

It can be 1 through 8 characters long.

The default is DFSYDRU0.

This parameter is optional; it is required if IBM MQ is to receive messages from an IMS application that was not started by IBM MQ. The name must correspond to the destination resolution user exit coded in the IMS system. For more information see "Using OTMA exits in IMS" on page 1575.

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.

`QSGDATA`
Queue-sharing group data. This keyword takes five positional parameters:

`QSGDATA=( Qsgname,Dsgname,Db2name,Db2serv ,Db2blob)`

Qsgname
This is the name of the queue-sharing group to which the queue manager belongs.

See Rules for naming IBM MQ objects for valid characters. The name:

- Can be 1 through 4 characters long
- Must not start with a numeric
- Must not end in @.

  This is because, for implementation reasons, names of less than four characters are padded internally with @ symbols,

The default is blanks, which indicates that the queue manager is not a member of any queue-sharing group.

Dsgname
This is the name of the Db2 data-sharing group to which the queue manager is to connect.

It can be 1 through 8 characters long and must be entered in uppercase characters.

The default is blanks, which indicates that you are not using queue-sharing groups.

Db2name
This is the name of the Db2 subsystem or group attachment to which the queue manager is to connect.

It can be 1 through 4 characters long and must be entered in uppercase characters.

The default is blanks, which indicates that you are not using queue-sharing groups.

Note: The Db2 subsystem (or group attachment) must be in the Db2 data-sharing group specified in the `Dsgname`, and all queue managers must specify the same Db2 data-sharing group.

Db2serv
This is the number of server tasks used for accessing Db2.

It can be in the range 4 through 10.

The default is 4.

Db2blob
This is the number of Db2 tasks used for accessing Binary Large Objects (BLOBs).

It can be in the range 4 through 10.

The default is 4.

If you specify one of the name parameters (that is, **`Qsgname`**, **`Dsgname`**, or **`Db2name`**), you must enter values for the other names, otherwise IBM MQ fails.

`RESAUDIT`
Specifies whether RACF audit records are written for RESLEVEL security checks performed during connection processing.

Specify one of:

NO RESLEVEL auditing is not performed.

YES RESLEVEL auditing is performed.

The default is YES.

`ROUTCDE`
Specifies the default z/OS message routing code assigned to messages that are not sent in direct response to an MQSC command.

Specify one of:

1. A value in the range 1 through 16, inclusive.

2. A list of values, separated by a comma and enclosed in parentheses. Each value must be in the range 1 through 16, inclusive.

The default is 1.

For more information about z/OS routing codes, see the *MVS Routing and Descriptor Codes* manual.

SERVICE
This field is reserved for use by IBM.

SMFACCT
Specifies whether IBM MQ sends accounting data to SMF automatically when the queue manager starts.

Specify one of:

NO Do not start gathering accounting data automatically.

YES Start gathering accounting data automatically for the default class 1.

integers
A list of classes for which accounting is gathered automatically in the range 1 through 4.

The default is NO.

SMFSTAT
Specifies whether to gather SMF statistics automatically when the queue manager starts.

Specify one of:

NO Do not start gathering statistics automatically.

YES Start gathering statistics automatically for the default class 1.

integers
A list of classes for which statistics are gathered automatically in the range 1 through 4.

The default is NO.

SPLCAP
The security policy capability enables higher level of message security through policies that control whether messages are signed, or encrypted, as they are written and read from queues.

Its use is licensed by a separately installed product, Advanced Message Security (AMS), which supplies an enabling module in the SDRQAUTH library.

Security policy processing is enabled for this queue manager, by configuring SPLCAP with one of the following values:

NO The capability to implement message security policies for queues is not enabled during queue manager initialization.

YES Message security capabilities are enabled during queue manager initialization.

If this control is set, the queue manager attempts to load the license enabling module from SDRQAUTH during initialization, and start an additional address space (AMSM).

The queue manager does not start unless AMS is licensed, and the necessary configuration for message security is in place.

The default is NO.

STATIME
Specifies the default time, in minutes, between consecutive gatherings of statistics.

Specify a number in the range zero through 1440.

If you specify a value of zero, both statistics data and accounting data is collected at the SMF data collection broadcast. See Using System Management Facility for information about setting this.

The default is 30.

TRACSTR

Specifies whether global tracing is to start automatically.

Specify one of:

NO Do not start global tracing automatically.

YES Start global tracing automatically for the default class, class 1.

integers
A list of classes for which global tracing is to be started automatically in the range 1 through 4.

* Start global trace automatically for all classes.

The default is NO if you do not specify the keyword in the macro.

Note: The supplied default system parameter load module (CSQZPARM) has TRACSTR=YES (set in the assembler module CSQFSYSP). If you do not want to start tracing automatically, either create your own system parameter module, or issue the STOP TRACE command after the queue manager has started.

For details about the STOP TRACE command, see STOP TRACE.

TRACTBL

Specifies the default size, in 4 KB blocks, of trace table where the global trace facility stores IBM MQ trace records.

Specify a value in the range 1 through 999.

The default is 99. This is equivalent to 396 KB.

Note: Storage for the trace table is allocated in the ECSA. Therefore, you must select this value with care.

WLMTIME

Specifies the time (in minutes or seconds depending on the value of WLMTIMU) between each scan of the indexes for WLM-managed queues.

Specify a value in the range 1 through 9999.

The default is 30.

WLMTIMU

Time units used with the WLMTIME parameter.

Specify one of :

MINS WLMTIME represents a number of minutes.

SECS WLMTIME represents a number of seconds.

The default is MINS.

Related reference:

"Using CSQ6LOGP"
Use this topic as a reference for how to specify logging options using CSQ6LOGP.

"Using CSQ6ARVP" on page 1501
Use this topic as a reference for how to specify your archiving environment using CSQ6ARVP

Using CSQ6LOGP: z/OS

Use this topic as a reference for how to specify logging options using CSQ6LOGP.

Use CSQ6LOGP to establish your logging options.

The default parameters for CSQ6LOGP, and whether you can alter each parameter using the SET LOG command, are shown in Default values of CSQ6LOGP parameters. If you need to change any of these values, refer to the detailed descriptions of the parameters.

Table 145. Default values of CSQ6LOGP parameters

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
V 9.0.0 ZHYWRITE	Specifies whether the zHyperWrite feature is enabled.	No	-

COMPLOG
Specifies whether log compression is enabled.

Specify either:

NONE
Log compression is not enabled.

RLE Log compression is enabled using run-length encoding.

ANY The queue manager selects the compression algorithm that gives the greatest degree of log record compression. This option results in RLE compression.

The default is NONE.

For more details about log compression, see Log compression.

DEALLCT
Specifies the length of time, in minutes, that an archive read tape unit is allowed to remain unused before it is deallocated.

Specify one of the following:

- Time, in minutes, in the range zero through 1440
- NOLIMIT

Specifying 1440 or NOLIMIT means that the tape unit is never deallocated.

The default is zero.

When archive log data is being read from tape, it is recommended that you set this value high enough to allow IBM MQ to optimize tape handling for multiple read applications.

INBUFF
Specifies the size, in kilobytes, of the input buffer for reading the active and archive logs during recovery. Use a decimal number in the range 28 through 60. The value specified is rounded up to a multiple of 4.

The default is 60 KB.

Suggested settings:

Test system	28 KB
Production system	60 KB Set this to the maximum for best log read performance.

MAXARCH
Specifies the maximum number of archive log volumes that can be recorded in the BSDS. When this number is exceeded, recording begins again at the start of the BSDS.

Use a decimal number in the range 10 through 1000.

The default is 500.

Suggested settings:

Test system	500 (default)
Production system	1 000 Set this to the maximum so that the BSDS can record as many logs as possible.

For information about the logs and BSDS, see Managing IBM MQ resources.

MAXCNOFF
Specifies the number of CSQJOFF7 offload tasks that can be run in parallel.

This allows a queue manager, or queue managers, to be tuned such that they will not use all the available tape units.

Instead the queue manager waits until a CSQJOFF7 offload task has completed before trying to allocate any new archive data sets.

If the queue manager is archiving to tape, set this parameter so that the number of concurrent tape requests should not equal, or exceed, the number of tape units available, otherwise the system might hang.

Note that if dual archiving is in use, then each offload task performs both archives, so the parameter needs to be set accordingly. For example if the queue manager is dual archiving to tape, a value of MAXCNOFF=2 would allow up to two active logs to be archived concurrently to four tapes.

If several queue managers are sharing the tape units, you should set the MAXCNOFF for each queue manager accordingly.

The default value is 31.

Specify a value in the range 1 through 31.

MAXRTU

Specifies the maximum number of dedicated tape units that can be allocated to read archive log tape volumes concurrently.

This parameter and the DEALLCT parameter allow IBM MQ to optimize archive log reading from tape devices.

Specify a value in the range 1 through 99.

The default is 2.

It is recommended that you set the value to be at least one less than the number of tape units available to IBM MQ. If you do otherwise, the offload process could be delayed, which could affect the performance of your system. For maximum throughput during archive log processing, specify the largest value possible for this option, remembering that you need at least one tape unit for offload processing.

OFFLOAD

Specifies whether archiving is on or off.

Specify either:

YES	Archiving is on
NO	Archiving is off

The default is YES.

Attention: Do **not** switch archiving off unless you are working in a test environment. If you do switch it off, you cannot guarantee that data will be recovered in the event of a system or transaction failure.

OUTBUFF

Specifies the total size, in kilobytes, of the storage to be used by IBM MQ for output buffers for writing the active and archive log data sets. Each output buffer is 4 KB.

The parameter must be in the range 80 through 4000. The value specified is rounded up to a multiple of 4. Values between 40and 80 will be accepted for compatibility reasons, and are treated as a value of 80.

The default is 4 000 KB.

Suggested settings:

Test system	400 KB
Production system	4 000 KB
	Set this value to the maximum to avoid running out of log output buffers.

TWOACTV

Specifies single or dual active logging.

Specify either:

NO	Single active logs
YES	Dual active logs

The default is YES.

For more information about the use of single and dual logging, see Managing IBM MQ resources.

TWOARCH

Specifies the number of archive logs that IBM MQ produces when the active log is offloaded.

Specify either:

NO	Single archive logs
YES	Dual archive logs

The default is YES.

Suggested settings:

Test system	NO
Production system	YES (default)

For more information about the use of single and dual logging, see Managing IBM MQ resources.

TWOBSDS

Specifies the number of bootstrap data sets.

Specify either:

NO	Single BSDS
YES	Dual BSDS

The default is YES.

For more information about the use of single and dual logging, see Managing IBM MQ resources.

WRTHRSH

Specifies the number of 4 KB output buffers to be filled before they are written to the active log data sets.

The larger the number of buffers, the less often the write takes place, and this improves the performance of IBM MQ. The buffers might be written before this number is reached if significant events, such as a commit point, occur.

Specify the number of buffers in the range 1 through 256.

The default is 20.

V 9.0.0 **ZHYWRITE**

Specifies whether the zHyperWrite feature is enabled.

The value can be:

NO zHyperWrite is not enabled.

Related reference:

"Using CSQ6SYSP" on page 1487
Use this topic as a reference for how to set system parameters using CSQ6SYSP.

"Using CSQ6ARVP"
Use this topic as a reference for how to specify your archiving environment using CSQ6ARVP

Using CSQ6ARVP: z/OS

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 146. If you need to change any of these values, refer to the detailed descriptions of the parameters. For more information about planning your archive storage, see Planning on z/OS.

Table 146. 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
UNIT	Device type or unit name on which the first copy of archive log data sets is stored.	TAPE	X
UNIT2	Device type or unit name on which the second copy of archive log data sets is stored.	Blank	X

ALCUNIT

Specifies the unit in which primary and secondary space allocations are made.

Specify one of:

CYL Cylinders

TRK Tracks

BLK Blocks

You are recommended to use BLK because it is independent of the device type.

The default is BLK.

If free space on the archive DASD volumes is likely to be fragmented, you are recommended to specify a smaller primary extent and allow expansion into secondary extents. For more information about space allocation for active logs, refer to the Planning on z/OS.

ARCPFX1
Specifies the prefix for the first archive log data set name.

See the TSTAMP parameter for a description of how the data sets are named and for restrictions on the length of ARCPFX1.

This parameter cannot be left blank.

The default is CSQARC1.

You might need to authorize the userid associated with the IBM MQ queue manager address space to create archive logs with this prefix.

ARCPFX2
Specifies the prefix for the second archive log data set name.

See the TSTAMP parameter for a description of how the data sets are named and for restrictions on the length of ARCPFX2.

This parameter cannot be blank even if the TWOARCH parameter is specified as NO.

The default is CSQARC2.

You might need to authorize the userid associated with the IBM MQ queue manager address space to create archive logs with this prefix.

ARCRETN
Specifies the retention period, in days, to be used when the archive log data set is created.

The parameter must be in the range zero through 9999.

The default is 9999.

Suggested settings:

Test system	3 In a test system, archive logs are probably not required over long periods.
Production system	9 999 (default) Set this value high to effectively switch automatic archive log deletion off.

For more information about discarding archive log data sets, see Administering IBM MQ for z/OS.

ARCWRTC
Specifies the list of z/OS routing codes for messages about the archive log data sets to the operator. This field is ignored if ARCWTOR is set to NO.

Specify up to 14 routing codes, each with a value in the range 1 through 16. You must specify at least one code. Separate codes in the list by commas, not by blanks.

The default is the list of values: 1,3,4.

For more information about z/OS routing codes, see the *MVS Routing and Descriptor Codes* manual.

ARCWTOR

Specifies whether a message is to be sent to the operator and a reply is received before attempting to mount an archive log data set.

Other IBM MQ users might be forced to wait until the data set is mounted, but they are not affected while IBM MQ is waiting for the reply to the message.

Specify either:

YES	The device needs a long time to mount archive log data sets. For example, a tape drive.
NO	The device does not have long delays. For example, DASD.

The default is YES.

Suggested settings:

Test system	NO
Production system	YES (default) This is dependent on operational procedures. If tape robots are used, NO might be more appropriate.

BLKSIZE

Specifies the block size of the archive log data set. The block size you specify must be compatible with the device type you specify in the UNIT parameter.

The parameter must be in the range 4 097 through 28 672. The value you specify is rounded up to a multiple of 4 096.

The default is 28 672.

This parameter is ignored for data sets that are managed by the storage management subsystem (SMS).

If the archive log data set is written to DASD, you are recommended to choose the maximum block size that allows 2 blocks for each track. For example, for a 3390 device, you should use a block size of 24 576.

If the archive log data set is written to tape, specifying the largest possible block size improves the speed of reading the archive log. You should use a block size of 28 672.

Suggested settings:

Test system	Use the block size recommendation depending on the media used for archive logs. That is, for disk 24 576, and tape 28 672.
Production system	Use the block size recommendation depending on the media used for archive logs. That is, for disk 24 576, and tape 28 672.

CATALOG

Specifies whether archive log data sets are cataloged in the primary integrated catalog facility (ICF) catalog.

Specify either:

NO	Archive log data sets are not cataloged
YES	Archive log data sets are cataloged

The default is NO.

All archive log data sets allocated on DASD must be cataloged. If you archive to DASD with the CATALOG parameter set to NO, message CSQJ072E is displayed each time an archive log data set is allocated, and IBM MQ catalogs the data set.

Suggested settings:

Test system	YES
Production system	YES, when archives are allocated on DASD

`COMPACT`

Specifies whether data written to archive logs is to be compacted. This option applies only to a 3480 or 3490 device that has the improved data recording capability (IDRC) feature. When this feature is turned on, hardware in the tape control unit writes data at a much higher density than normal, allowing for more data on each volume. Specify NO if you do not use a 3480 device with the IDRC feature or a 3490 base model, except for the 3490E. Specify YES if you want the data to be compacted.

Specify either:

NO	Do not compact the data sets
YES	Compact the data sets

The default is NO.

Specifying YES adversely affects performance. Also be aware that data compressed to tape can be read only using a device that supports the IDRC feature. This can be a concern if you have to send archive tapes to another site for remote recovery.

Suggested settings:

Test system	Not applicable
Production system	NO (default) This applies to 3480 and 3490 IDR compression only. Setting this to YES might degrade archive log read performance during recovery and restart; however, it does not affect writing to tape.

`PRIQTY`

Specifies the primary space allocation for DASD data sets in ALCUNITs.

The value must be greater than zero.

The default is 25 715.

This value must be sufficient for a copy of either the log data set or its corresponding BSDS, whichever is the larger. To determine the necessary value, follow this procedure:

1. Determine the number of active log records allocated (`c`) as explained in “Task 14: Create the bootstrap and log data sets” on page 1482.
2. Determine the number of 4096 byte blocks in each archive log block:

```
d = BLKSIZE / 4096
```

   where BLKSIZE is the rounded up value.
3. If ALCUNIT=BLK:

```
PRIQTY = INT(c / d) + 1
```

where INT means round down to an integer.

If ALCUNIT=TRK:

```
PRIQTY = INT(c / (d * INT(e/BLKSIZE))) + 1
```

where e is the number of bytes for each track (56664 for a 3390 device) and INT means round down to an integer.

If ALCUNIT=CYL:

```
PRIQTY = INT(c / (d * INT(e/BLKSIZE) * f)) + 1
```

where f is the number of tracks for each cylinder (15 for a 3390 device) and INT means round down to an integer.

For information about how large to make your log and archive data sets, see "Task 14: Create the bootstrap and log data sets" on page 1482 and "Task 15: Define your page sets" on page 1484.

Suggested settings:

Test system	1 680 Sufficient to hold the entire active log, that is: `10 080 / 6 = 1 680 blocks`
Production system	Not applicable when archiving to tape.

If free space on the archive DASD volumes is likely to be fragmented, you are recommended to specify a smaller primary extent and allow expansion into secondary extents. For more information about space allocation for active logs, refer to the Planning on z/OS.

PROTECT

Specifies whether archive log data sets are to be protected by discrete ESM (external security manager) profiles when the data sets are created.

Specify either:

NO Profiles are not created.

YES Discrete data set profiles are created when logs are offloaded. If you specify YES:
- ESM protection must be active for IBM MQ.
- The user ID associated with the IBM MQ queue manager address space must have authority to create these profiles.
- The TAPEVOL class must be active if you are archiving to tape.

Otherwise, offloading fails.

The default is NO.

QUIESCE

Specifies the maximum time in seconds allowed for the quiesce when an ARCHIVE LOG command is issued with MODE(QUIESCE) specified.

The parameter must be in the range 1 through 999.

The default is 5.

SECQTY

Specifies the secondary space allocation for DASD data sets in ALCUNITs. The secondary extent can be allocated up to 15 times; see the *z/OS MVS JCL Reference* and *z/OS MVS JCL User's Guide* for details.

The parameter must be greater than zero.

The default is 540.

TSTAMP

Specifies whether the archive log data set name has a time stamp in it.

Specify either:

NO Names do not include a time stamp. The archive log data sets are named:

```
arcpfxi.A nnnnnnn
```

Where *arcpfxi* is the data set name prefix specified by ARCPFX1 or ARCPFX2. *arcpfxi* can have up to 35 characters.

YES Names include a time stamp. The archive log data sets are named:

```
arcpfxi.cyyddd.T hhmmsst.A nnnnnnn
```

where *c* is 'D' for the years up to and including 1999 or 'E' for the year 2000 and later, and *arcpfxi* is the data set name prefix specified by ARCPFX1 or ARCPFX2. *arcpfxi* can have up to 19 characters.

EXT Names include a time stamp. The archive log data sets are named:

```
arcpfxi.D yyyyddd.T hhmmsst.A nnnnnnn
```

Where *arcpfxi* is the data set name prefix specified by ARCPFX1 or ARCPFX2. *arcpfxi* can have up to 17 characters.

The default is NO.

UNIT

Specifies the device type or unit name of the device that is used to store the first copy of the archive log data set.

Specify a device type or unit name of 1 through 8 alphanumeric characters. The first character must be alphabetic.

This parameter cannot be blank.

The default is TAPE.

If you archive to DASD, you can specify a generic device type with a limited volume range, for example, UNIT=3390.

If you archive to DASD, make sure that:

- The primary space allocation is large enough to contain all the data from the active log data sets.
- The archive log data set catalog option (CATALOG) is set to YES.
- You have used a proper value for BLKSIZE.

If you archive to TAPE, IBM MQ can extend to a maximum of 20 volumes.

Suggested settings:

Test system	DASD
Production system	TAPE

For more information about choosing a location for archive logs, refer to the Planning on z/OS.

UNIT2
Specifies the device type or unit name of the device that is used to store the second copy of the archive log data sets.

Specify a device type or unit name of 1 through 8 alphanumeric characters. The first character must be alphabetic. If this parameter is blank, the value set for the UNIT parameter is used.

The default is blank.

Related reference:
"Using CSQ6SYSP" on page 1487
Use this topic as a reference for how to set system parameters using CSQ6SYSP.
"Using CSQ6LOGP" on page 1497
Use this topic as a reference for how to specify logging options using CSQ6LOGP.

Using CSQ6USGP: z/OS V 9.0.3

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 147. 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 147. 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	Blank

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 `5655⌂MQ9`.

VUE Queue manager usage is recorded as a stand-alone IBM MQ for z/OS Value Unit Edition (VUE) product, with product ID `5655⌂VU9`.

ADVANCEDVUE
Queue manager usage is recorded as part of an IBM MQ Advanced for z/OS, Value Unit Edition product, with product ID `5655⌂AV1`.

AMSPROD
Specifies the product against which AMS usage is to be recorded, if used.

Specify one of:

AMS AMS usage is recorded as a stand-alone Advanced Message Security for z/OS product, with product ID `5655⌂AM9`.

ADVANCED
: AMS usage is recorded as part of an IBM MQ Advanced for z/OS product, with product ID `5655⌂AV9`.

ADVANCEDVUE
: AMS usage is recorded as part of an IBM MQ Advanced for z/OS, Value Unit Edition product, with product ID `5655⌂AV1`.

See z/OS MVS Product Management for more information on product usage recording.

Related reference:

"Using CSQ6SYSP" on page 1487
Use this topic as a reference for how to set system parameters using CSQ6SYSP.

"Using CSQ6LOGP" on page 1497
Use this topic as a reference for how to specify logging options using CSQ6LOGP.

Task 18: Tailor the channel initiator parameters

z/OS

Use ALTER QMGR to customize the channel initiator to suit your requirements.

- *Repeat this task for each IBM MQ queue manager, as required.*
- *You must perform this task when migrating from a previous version.*

A number of queue manager attributes control how distributed queuing operates. Set these attributes using the MQSC command ALTER QMGR. The initialization data set sample thlqual.SCSQPROC(CSQ4INYG) contains some settings that you can customize. For more information, see ALTER QMGR.

The values of these parameters are displayed as a series of messages each time you start the channel initiator.

The relationship between adapters, dispatchers, and maximum number of channels

The ALTER QMGR parameters CHIADAPS and CHIDISPS define the number of task control blocks (TCBs) used by the channel initiator. CHIADAPS (adapter) TCBs are used to make IBM MQ API calls to the queue manager. CHIDISPS (dispatcher) TCBs are used to make calls to the communications network.

The ALTER QMGR parameter MAXCHL influences the distribution of channels over the dispatcher TCBs.

CHIDISPS

If you have a small number of channels use the default value.

One task for each processor optimizes system performance. As dispatcher tasks are CPU intensive, the principle is to keep as few tasks as busy as possible, so that the time taken to find and start threads is minimized.

CHIDISPS(20) is suitable for systems with more than 100 channels. There is unlikely to be any significant disadvantage in having CHIDISPS(20) where this is more dispatcher TCBs than necessary.

As a guideline, if you have more than 1000 channels, allow one dispatcher for every 50 current channels. For example, specify CHIDISPS(40) to handle up to 2000 active channels.

If you are using TCP/IP, the maximum number of dispatchers used for TCP/IP channels is 100, even if you specify a larger value in CHIDISPS.

CHIADAPS

Each IBM MQ API call to the queue manager is independent of any other and can be made on any adapter TCB. Calls using persistent messages can take much longer than those for nonpersistent messages because of log I/O. Thus a channel initiator processing a large number of persistent messages across many channels may need more than the default 8 adapter TCBs for optimum performance. This is particularly so where achieved batchsize is small, because end of batch processing also requires log I/O, and where thin client channels are used.

The suggested value for a production environment is CHIADAPS(30). Using more than this is unlikely to give any significant extra benefit, and there is unlikely to be any significant disadvantage in having CHIADAPS(30) if this is more adapter TCBs than necessary.

MAXCHL

Each channel is associated with a particular dispatcher TCB at channel start and remains associated with that TCB until the channel stops. Many channels can share each TCB. MAXCHL is used to spread channels across the available dispatcher TCBs. The first (MIN((MAXCHL / CHIDISPS) , 10)) channels to start are associated with the first dispatcher TCB, and so on, until all dispatcher TCBs are in use.

The effect of this for small numbers of channels and a large MAXCHL is that channels are NOT evenly distributed across dispatchers. For example, if you set CHIDISPS(10) and left MAXCHL at its default value of 200 but had only 50 channels, five dispatchers would be associated with 10 channels each and five would be unused. We suggest setting MAXCHL to the number of channels actually to be used where this is a small fixed number.

If you change this queue manager property, you must also review the ACTCHL, LU62CHL, and TCPCHL queue manager properties to ensure that the values are compatible. See Queue manager parameters for a full description of these properties, and their relationship.

Setting up your z/OS UNIX System Services environment for channel initiators

The channel initiator (CHINIT) uses OMVS threads. Review the OMVS configuration parameters before creating a new CHINIT, or modifying the number of dispatchers or SSLTASKS.

Each CHINIT uses 3 + CHIDISP + SSLTASKS OMVS threads. These contribute to the total number of OMVS threads used in the LPAR, and towards the number of threads used by CHINIT started task user ID.

You can use the **`D OMVS,L`** and review the current usage, highwater usage, and system limit of MAXPROCSYS (the maximum number of processes that the system allows).

If you are adding a new CHINIT or increasing the values of CHIDISPS or SSLTASKS then you must calculate the increase in threads and review the impact on the MAXPROCSYS values. You can use the **`SETOMVS`** command to dynamically change the MAXPROCSYS, or update the BPXPRCxx parmlib value or both.

The OMVS parameter MAXPROCUSER is the number of OMVS threads a single OMVS user, that is with the same UID, can have. The threads count towards this value. So if you have 2 CHINITS with the same started task user ID, with 10 dispatchers and 3 SSLTASKS each then there are 2 *(3 +10 + 3) = 32 threads for the OMVS uid.

You can display the default MAXPROCUSER by issuing the **`D OMVS,O`** command and you can use the **`SETOMVS`** command to dynamically change the MAXPROCUSER, or update the BPXPRCxx parmlib value or both.

You can override this value on a per user basis with the RACF command **`ALTUSER userid OMVS(PROCUSERMAX(nnnn))`** or equivalent.

To start the channel initiator, issue the following command:

```
START CHINIT
```

To ensure that the channel initiator has started successfully, check that there is no ICH408I error in the xxxxCHIN(ssidCHIN) job log.

Related concepts:

"Task 19: Set up Batch, TSO, and RRS adapters"
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 information:

Channel initiator statistics data records

Task 19: Set up Batch, TSO, and RRS adapters

z/OS

Make the adapters available to applications by adding libraries to appropriate STEPLIB concatenations. To cater for SNAP dumps issued by an adapter, allocate a CSQSNAP DDname. Consider using CSQBDEFV to improve the portability of your application programs

- *Repeat this task for each IBM MQ queue manager as required.*
- *You might need to perform this task when migrating from a previous version.*

To make the adapters available to batch and other applications using batch connections, add the following IBM MQ libraries to the STEPLIB concatenation for your batch application :

- thlqual.SCSQANL *x*
- thlqual.SCSQAUTH

where *x* is the language letter for your national language. (You do not need to do this if the libraries are in the LPA or the link list.)

For TSO applications add the libraries to the STEPLIB concatenation in the TSO logon procedure or activate them using the TSO command TSOLIB.

If the adapter detects an unexpected IBM MQ error, it issues an z/OS SNAP dump to DDname CSQSNAP, and issues reason code `MQRC_UNEXPECTED_ERROR` to the application. If the CSQSNAP DD statement is not in the application JCL or CSQSNAP is not allocated to a data set under TSO, no dump is taken. If this happens, you could include the CSQSNAP DD statement in the application JCL or allocate CSQSNAP to a data set under TSO and rerun the application. However, because some problems are intermittent, it is recommended that you include a CSQSNAP statement in the application JCL or allocate CSQSNAP to a data set in the TSO logon procedure to capture the reason for failure at the time it occurs.

The supplied program CSQBDEFV improves the portability of your application programs. In CSQBDEFV, you can specify the name of a queue manager, or queue-sharing group, to be connected to rather than specifying it in the `MQCONN` or `MQCONNX` call in an application program. You can create a new version of CSQBDEFV for each queue manager, or queue-sharing group. To do this, follow these steps:

1. Copy the IBM MQ assembler program CSQBDEFV from thlqual.SCSQASMS to a user library.
2. The supplied program contains the default subsystem name CSQ1. You can retain this name for testing and installation verification. For production subsystems, you can change the NAME=CSQ1 to your one- to four-character subsystem name, or use CSQ1.

   If you are using queue-sharing groups, you can specify a queue-sharing group name instead of CSQ1. If you do this, the program issues a connect request to an active queue manager within that group.
3. Assemble and link-edit the program to produce the CSQBDEFV load module. For the assembly, include the library thlqual.SCSQMACS in your SYSLIB concatenation; use the link-edit parameters

`RENT,AMODE=31,RMODE=ANY`. This is shown in the sample JCL in thlqual.SCSQPROC(CSQ4DEFV). Then include the load library in the z/OS Batch or the TSO STEPLIB, ahead of thlqual.SCSQAUTH.

Related concepts:

"Task 20: Set up the operations and control panels"
To set up the operations and control panels you must first set up the libraries that contain the required panels, EXECs, messages, and tables. To do this, you must take into account which national language feature is to be used for the panels. When you have done this, you can optionally update the main ISPF menu for IBM MQ operations and control panels and change the function key settings.

Task 20: Set up the operations and control panels

z/OS

To set up the operations and control panels you must first set up the libraries that contain the required panels, EXECs, messages, and tables. To do this, you must take into account which national language feature is to be used for the panels. When you have done this, you can optionally update the main ISPF menu for IBM MQ operations and control panels and change the function key settings.

- *You need to perform this task once for each z/OS system where you want to run IBM MQ.*
- *You might need to perform this task when migrating from a previous version.*

Setting up the libraries

Follow these steps to set up the IBM MQ operations and control panels:

1. Ensure that all the libraries contained in your concatenations are either in the same format (F, FB, V, VB) and have the same block size, or are in order of decreasing block sizes. Otherwise, you might have problems trying to use these panels.
2. Include the library thlqual.SCSQEXEC in your SYSEXEC or SYSPROC concatenation or activate it using the TSO ALTLIB command. This library, which is allocated with a fixed-block 80 record format during installation, contains the required EXECs.

   It is preferable to put the library into your SYSEXEC concatenation. However, if you want to put it in SYSPROC, the library must have a record length of 80 bytes.
3. Add thlqual.SCSQAUTH and thlqual.SCSQANLx to the TSO logon procedure STEPLIB or activate it using the TSO TSOLIB command, if it is not in the link list or the LPA.
4. You can either add the IBM MQ panel libraries permanently to your ISPF library setup, or allow them to be set up dynamically when the panels are used. For the former choice, you need to do the following:
   a. Include the library containing the operations and control panel definitions in your ISPPLIB concatenation. The name is thlqual.SCSQPNLx, where x is the language letter for your national language.
   b. Include the library containing the required tables in your ISPTLIB concatenation. The name is thlqual.SCSQTBLx, where x is the language letter for your national language.
   c. Include the library containing the required messages in your ISPMLIB concatenation. The name is thlqual.SCSQMSGx, where x is the language letter for your national language.
   d. Include the library containing the required load modules in your ISPLLIB concatenation. The name of this library is thlqual.SCSQAUTH.
5. Test that you can access the IBM MQ panels from the TSO Command Processor panel. This is usually option 6 on the ISPF/PDF Primary Options Menu. The name of the EXEC that you run is CSQOREXX. There are no parameters to specify if you have put the IBM MQ libraries permanently in your ISPF setup as in step 4. If you have not, use the following:

```
CSQOREXX thlqual langletter
```

where `langletter` is a letter identifying the national language to be used:

C	Simplified Chinese
E	U.S. English (mixed case)
F	French
K	Japanese
U	U.S. English (uppercase)

Updating the ISPF menu

You can update the ISPF main menu to allow access to the IBM MQ operations and control panels from ISPF. The required setting for &ZSEL is:

```
CMD(%CSQOREXX thlqual langletter)
```

For information about `thlqual` and `langletter`, see Step 5 on page 1511.

For more details, see the *z/OS: ISPF Dialog Developer's Guide and Reference* manual.

Updating the function keys and command settings

You can use the normal ISPF procedures for changing the function keys and command settings used by the panels. The application identifier is CSQO.

However, this is not recommended because the help information is not updated to reflect any changes that you have made.
Related concepts:
"Task 21: Include the IBM MQ dump formatting member"
To be able to format IBM MQ dumps using the Interactive Problem Control System (IPCS), you must update some system libraries.

Task 21: Include the IBM MQ dump formatting member

z/OS

To be able to format IBM MQ dumps using the Interactive Problem Control System (IPCS), you must update some system libraries.
- *You need to perform this task once for each z/OS system where you want to run IBM MQ.*
- *You need to perform this task when migrating from a previous version.*

To be able to format IBM MQ dumps using the Interactive Problem Control System (IPCS), copy the data set thlqual.SCSQPROC(CSQ7IPCS) to SYS1.PARMLIB. You should not need to edit this data set.

If you have customized the TSO procedure for IPCS, thlqual.SCSQPROC(CSQ7IPCS) can be copied into any library in the IPCSPARM definition. See the *MVS IPCS Customization* manual for details on IPCSPARM.

You must also include the library thlqual.SCSQPNLA in your ISPPLIB concatenation.

To make the dump formatting programs available to your TSO session or IPCS job, you must also include the library thlqual.SCSQAUTH in your STEPLIB concatenation or activate it using the TSO TSOLIB command (even if it is already in the link list or LPA).

Related concepts:

"Task 22: Suppress information messages"
Your IBM MQ system might produce a large number of information messages. You can prevent selected messages being sent to the console or to the hardcopy log.

Task 22: Suppress information messages

z/OS

Your IBM MQ system might produce a large number of information messages. You can prevent selected messages being sent to the console or to the hardcopy log.

- *You need to perform this task once for each z/OS system where you want to run IBM MQ.*
- *You do not need to perform this task when migrating from a previous version.*

If your IBM MQ system is heavily used, with many channels stopping and starting, a large number of information messages are sent to the z/OS console and hardcopy log. The IBM MQ - IMS bridge and buffer manager might also produce a large number of information messages.

If required, you can suppress some of these console messages by using the z/OS message processing facility list, specified by the MPFLSTxx members of SYS1.PARMLIB. The messages you specify still appear on the hardcopy log, but not on the console.

Sample thlqual.SCSQPROC(CSQ4MPFL) shows suggested settings for MPFLSTxx. See the *MVS Initialization and Tuning Reference* manual for more information about MPFLSTxx.

If you want to suppress selected information messages on the hardcopy log, you can use the z/OS installation exit IEAVMXIT. You can set the following bit switches ON for the required messages:

CTXTRDTM
Delete the message.

The message is not displayed on consoles or logged in hardcopy.

CTXTESJL
Suppress from job log.

The message does not go into the JES job log.

CTXTNWTP
Do not carry out WTP processing.

The message is not sent to a TSO terminal or to the system message data set of a batch job.

Note:

1. For full details on the other parameters, refer to the MVS Installation Exits documentation.
2. You are not recommended to suppress messages other than those in the suggested suppression list, CSQ4MPFL.

In addition you can specify the extra parameter:

EXCLMSG
Specifies a list of messages to be excluded from any log.

Messages in this list are not sent to the z/OS console and hardcopy log. See EXCLMSG in "Using CSQ6SYSP" on page 1487 for further information.

Related tasks:
"Testing a queue manager on z/OS" on page 1521
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.

Task 23: Create procedures for Advanced Message Security

z/OS

Each IBM MQ subsystem that is to be configured to use Advanced Message Security requires a cataloged procedure to start the AMS address space. You can create your own or use the IBM-supplied procedure library.

For each IBM MQ subsystem that is to be configured to use Advanced Message Security tailor a copy of sample procedure CSQ4AMSM. To do this, perform the following steps:
1. Copy the sample started task procedure *thlqual*.SCSQPROC(CSQ4AMSM) to your SYS1.PROCLIB or, if you are not using SYS1.PROCLIB, your procedure library. Name the procedure xxxxAMSM, where xxxx is the name of your IBM MQ subsystem. For example, CSQ1AMSM would be the AMS started task procedure for queue manager CSQ1.
2. Make a copy for each IBM MQ subsystem that you are going to use.
3. Tailor the procedures to your requirements using the instructions in the sample procedure CSQ4AMSM. You can also use symbolic parameters in the JCL to allow the procedure to be modified when it is started.
4. Review and optionally change the parameters passed to the AMS task using the Language Environment® _CEE_ENVFILE file. The sample thlqual.SCSQPROC(CSQ40ENV) lists the supported parameters.

Note: This task should be repeated for each IBM MQ queue manager.

Task 24: Set up the started task user Advanced Message Security

z/OS

The Advanced Message Security task requires a user ID that allows it to be known as a UNIX System Services process.

In addition, the users that the task works on behalf of must also have an appropriate definition of a UNIX UID (user ID) and GID (group ID) so these users are known as UNIX System Services users. For more information on defining UNIX System Services UIDs and GIDs, see *z/OS: Security Server RACF Security Administrator's Guide*.

z/OS: UNIX System Services Planning compares traditional UNIX security to z/OS security. The primary difference between traditional UNIX security and z/OS security is that the Kernel services support two levels of appropriate privileges: UNIX level and z/OS UNIX level.

Depending on your installation's security policy, the 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.

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 UNIX System Services processes.

For reference, the publications useful to this review are:
- *z/OS: UNIX System Services Planning*.
- *z/OS: Security Server RACF Security Administrator's Guide*.

Note: Choose the user ID for this task carefully because the 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*.

1. First define RACF user profiles for the Advanced Message Security started task user. These can be the same user.

   ```
   ADDUSER WMQAMSM NAME(' Advanced Message Security user') OMVS (UID(0)) DFLTGRP(group)
   ```

   Select a default 'group' as appropriate to your installation standards.

   Note: If you do not want to grant USS superuser authority (UID(0)), then you must permit the 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 IDs you selected or created in step 1:

   ```
   RDEFINE STARTED qmgr AMSM.* STDATA(USER(WMQAMSM))
   ```

   where *qmgr* is the name of prefix of the started task name. For example, the started tasks may be named CSQ1AMSM. In this case, you would substitute `qmgr AMSM.*` with `CSQ1AMSM.*`.

   The started task names must be named `qmgr AMSM.*`.
4. Use the SETROPTS RACF command to refresh the in-storage RACLISTed started class profiles:

   ```
   SETROPTS RACLIST(STARTED) REFRESH
   ```
5. The Advanced Message Security task temporarily assumes the identity of the host user ID of the client requestor during protection processing of IBM MQ messages. Therefore, it is necessary to define profiles in the SURROGAT class for each user ID that can make requests.

This can be done with a single generic profile if the RACF SURROGAT class is active. The check is ignored if the SURROGAT class is not active. The SURROGAT profiles needed are described in *z/OS: UNIX System Services Planning*.

To define profiles in the SURROGAT class:

a. Activate the RACF SURROGAT class using the RACF SETROPTS command:

```
SETROPTS CLASSACT(SURROGAT)
```

b. Activate generic profile processing for the RACF SURROGAT class:

```
SETROPTS GENERIC(SURROGAT)
```

c. Activate generic profile command processing for the RACF SURROGAT class:

```
SETROPTS GENCMD(SURROGAT)
```

d. Define a surrogate class generic profile:

```
RDEFINE SURROGAT BPX.SRV.* UACC(NONE)
```

e. Permit the Advanced Message Security user ID to the generic SURROGAT class profile:

```
PERMIT BPX.SRV.* CLASS(SURROGAT) ID(WMQAMSM) ACCESS(UPDATE)
```

Note: You can define more specific profiles if you want to restrict specific users to be processed by the Advanced Message Security task, as described in *z/OS: UNIX System Services Planning*.

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)
```

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.LISTRING CLASS(FACILITY) ID(WMQAMSM) ACC(READ)
   ```

   Alternatively, you can grant READ access to the data service task user's keyring in the RDATALIB class as follows:

   ```
   PERMIT WMQASMD.DRQ.AMS.KEYRING.LST CLASS(RDATALIB) ID(WMQAMSM) ACC(READ)
   ```

Resource security for AMS

The started task user requires read authority to the SYSTEM.PROTECTION.POLICY.QUEUE.

The started task user requires authority to connect to the queue manager as a BATCH application. For further information, see Connection security profiles for batch connections.

Task 25: Grant RACDCERT permissions to the security administrator for Advanced Message Security

▶ z/OS

Your Advanced Message Security security administrator requires authority to use the RACDCERT command to create and manage digital certificates.

Identify the appropriate user ID for this role and grant permission to use the RACDCERT command. For example:

```
PERMIT IRR.DIGTCERT.* CLASS(FACILITY) ID(admin) ACCESS(CONTROL)
SETROPTS RACLIST(FACILITY) REFRESH
```

where `admin` is the user ID of your Advanced Message Security security administrator.

Task 26: Grant users resource permissions for Advanced Message Security

▶ z/OS

Advanced Message Security users require relevant resource permissions.

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 RDATALIB
- Permissions for ICSF class CSFSERV and CSFKEYS profiles

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 may choose to use this default rather than explicitly defining the OMVS segment for each user.

The *z/OS: Security Server RACF Security Administrator's Guide* contains the detailed procedure for defining default OMVS segments. Review the procedure as outlined in this publication to determine if the definition of default OMVS segments in RACF User and Group profiles is appropriate to your installation.

To grant READ permission to the IRR.DIGTCERT.LISTRING class facility to all Advanced Message Security users, issue this command:

```
RDEFINE FACILITY IRR.DIGTCERT.LISTRING UACC(READ)
```

or grant READ permission on a per user basis by issuing this command:

```
PERMIT IRR.DIGTCERT.LISTRING CLASS(FACILITY) ID(userid) ACCESS(READ)
```

where userid is the name of the Advanced Message Security user.

Alternatively, you can use the RDATALIB class to grant access to specific keyrings (the RDATALIB permissions take precedence over IRR.DIGTCERT.LISTRING permissions). For example:

```
PERMIT user.DRQ.AMS.KEYRING.LST CLASS(RDATALIB) ID(user) ACC(READ)
```

If you are using ICSF-managed certificates and private keys, Advanced Message Security users require access to certain class CSFSERV and CSFKEYS profiles. This access is detailed in the following table:

Table 148. Required user access to class CSFSERV and CSFKEYS profiles

Class	Profile	Permission
CSFSERV	CSFDSG	READ
CSFSERV	CSFPKE	READ
CSFSERV	CSFPKD	READ
CSFSERV	CSFDSV	READ
CSFKEYS	ICSF PKDS Label	READ

Create the Liberty server definition

z/OS V 9.0.1

If you installed the IBM MQ for z/OS Unix System Services Web Components, and want to use the MQ Console, or the REST API, you need to create and customize the Liberty server definition.

Before you begin

You need to create the SYSTEM.REST.REPLY.QUEUE in order to use the Liberty server. Do this by using the latest **CSQ4INSG** sample in “Task 13: Customize the initialization input data sets” on page 1479.

About this task

- You need to perform this task once for each z/OS system where you want to run the MQ Console or REST API.
- You need a Liberty server for each version of IBM MQ that is running.
- You might need to refresh or modify the server configuration when migrating from a previous version.

IBM MQ for z/OS Unix System Services Web Components requires the creation of a single Liberty server, called `mqweb`.

The server configuration and log files are all stored under the Liberty user directory.

Carry out the following procedure to create the `mqweb` server definition:

Procedure

1. 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 will contain log files, as well as the server configuration, you should create this directory in a separate file system.
2. Ensure that your current directory is `PathPrefix/web/bin`, which is the location of the **`crtmqweb.sh`** script. `PathPrefix` is the IBM MQ Unix System Services Components installation path.
3. Create the Liberty user directory, containing the template `mqweb` server definition, by running the **`crtmqweb.sh`** script.

   Note: The **`crtmqweb.sh`** script accepts one optional parameter - the name of the Liberty user directory.

   If you do not supply a name for the Liberty user directory, a default value of `/var/mqm/web/installation1` is used.
4. 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, 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

Return to Configuring the IBM MQ Console on z/OS without security and follow the procedure from Step 2 onwards.

Create a procedure for the Liberty server

z/OS V 9.0.1

If you installed the IBM MQ for z/OS Unix System Services Web Components, and want to use the MQ Console, or the REST API, you need to create a cataloged procedure to start the Liberty `mqweb` server.

- You need to perform this task once for each z/OS system where you want to run IBM MQ.
- You need a Liberty server instance for each version of IBM MQ that is running. For example, a started task called MQWB0901 for queue managers at Version 9.0.1 and a started task called MQWB0902 for queue managers at Version 9.0.2.

  If you have only one queue manager, 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 `thlqual.SCSQPROC(CSQ4WEBS)` to your procedure library.

   Name the procedure according to the standards of your enterprise.

   For example `MQWB0901`, indicating that this is the cataloged procedure for Liberty for IBM MQ Version 9.0.1
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.sh`** script was run to create the `mqweb` server definition.

   See “Create the Liberty server definition” on page 1518 for details.
3. Authorize the procedure to run under your external security manager.
4. Use the **`S procname`** command to start the procedure.

   This should produce message +CWWKE0001I: The server mqweb has been launched.

   If the server does not start successfully, review the messages.

   When the procedure starts, the output is stored in files under the USERDIR parameter. For example, if the user directory is `/u/mq/mqweb`, check `/u/mq/mqweb/servers/mqweb/logs`.

   The files are written in ASCII, so you can use your normal system tools to view the files.
5. Use IBM Workload Manager (WLM) to classify this address space.

   The Liberty 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.
6. use the **`P procname`** command to stop the procedure.

   Notes:

   a. Ensure that you specify **Caps off** when you edit the member, as the file has lower case data.
   b. The web server can take a considerable time to start up or shut down, for example, more than a minute.

What to do next

Task 30: Configure IBM MQ Console security

Configuring the IBM MQ Console and REST API

V 9.0.1

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

- "Configuring security" on page 1398
- "Configuring the HTTP host name" on page 1399
- "Configuring the HTTP and HTTPS ports" on page 1401
- "Configuring the response timeout" on page 1402
- "Configuring autostart" on page 1403
- "Configuring logging" on page 1405
- "Configuring the LTPA token expiry interval" on page 1407
- "Configuring the messaging REST API" on page 1409
- "Configuring CSRF protection" on page 1398

Configuring the `ReportingService` (formerly `BluemixRegistration`) stanza

z/OS V 9.0.3 MQ Adv. VUE

You can carry out this process only if your enterprise is using IBM MQ Advanced for z/OS, Value Unit Edition. If you want to publish registration and usage data for your queue manager to the IBM Cloud Product Insights service on IBM Cloud (formerly Bluemix), you need to perform this task once, for each queue manager.

See: IBM Cloud Product Insights service for more information.

Create a member in the customized configuration dataset for the queue manager called CSQMQMIN.

The CSQ4QMIN sample can be copied from the `thlqual.SCSQPROC` library. The CSQMQMIN contains configuration information to configure the `ReportingService` stanza in it, as described in the following text: V 9.0.5

```
ReportingService:
 APIKey = apikey
 ServiceURL = service_url
       Serviceproxy=optional_value_for_the_proxy_to_the_BlueMix_service
```

where

APIKey
: Is the IBM Cloud Product Insights API public key to use when logging on to the Product Insights service.

  The **`APIKey`** should not be enclosed within quotation marks.

  The public key is returned when creating an instance of the Product Insights service on IBM Cloud. See the IBM Cloud Product Insights developer center for further information about creating service instances.

ServiceURL
: Is the URL of the Product Insights service. `/api/startup` or `/api/usage` are appended to the URL when publishing data to the service.

The **ServiceURL** is the API host given when the service is IBM Cloud.

V 9.0.4 **ServiceProxy**
An optional value for the proxy to the IBM Cloud service.

Notes:

1. The data set can be specified in character code pages 37 or 1047.
2. The record size for the data set must be set to FB 80.
3. There should be no line sequence numbers in the data set.
4. Stanza names in the data set should appear on one line.
5. If required, the plus sign (+) can be used as a continuation character when specifying key value pairs. The plus sign can appear in any column, up to column 80.
6. The data set is read during queue manager start-up.
7. If the queue manager fails to parse the data set, the queue manager issues suitable console error messages, and continues processing.

V 9.0.4 Sample `CSQ4QMIN`, in the `thlqual.SCSQPROC` library, has been provided to help you with defining this data set.

Once you have defined the data set, you need to name the data set on the CSQMQMIN DD card in your queue manager start-up procedure, and uncomment the DD card. See Create procedures for the IBM MQ queue manager for more details.

Related tasks:

"Configuring IBM MQ Advanced for z/OS VUE for use with IBM Cloud Product Insights service in IBM Cloud (formerly Bluemix)" on page 1592
Connect your IBM MQ Advanced for z/OS, Value Unit Edition queue manager to the IBM Cloud Product Insights service in IBM Cloud (formerly Bluemix) to report and view startup and usage information.

"Configuring a z/OS queue manager for use with the IBM Cloud Product Insights service instance on IBM Cloud (formerly Bluemix)" on page 1595
You can carry out this process only if your enterprise is using IBM MQ Advanced for z/OS, Value Unit Edition. Your IBM Cloud Product Insights service instance dashboard shows data for only the queue managers that are configured to include the security and IBM Cloud registration information.

Testing a queue manager on z/OS

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"
- "Testing for queue-sharing groups" on page 1526
- "Testing for distributed queuing" on page 1527
- "Testing for C, C++, COBOL, PL/I, and CICS programs with IBM MQ for z/OS" on page 1531

Related tasks:

"Configuring queue managers on z/OS" on page 1455
Use these instructions to configure queue managers on IBM MQ for z/OS.

Related information:

IBM MQ for z/OS concepts

Planning your IBM MQ environment on z/OS

Administering IBM MQ for z/OS

Running the basic installation verification program

z/OS

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 177. 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 177. 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 178 on page 1526

z/OS 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 QUALIFER : CSQ4IVP1
INSTALLATION VERIFICATION BEGINS :
STAGE 01 COMPLETE. COMPCODE : 0000 REASON CODE : 0000
STAGE 02 INFO: QMGR EVENT SWITCH IS OFF FOR BRIDGE EVENTS
STAGE 02 INFO: QMGR EVENT SWITCH IS EXCP FOR CHANNEL EVENTS
STAGE 02 INFO: QMGR EVENT SWITCH IS OFF FOR SSL EVENTS
STAGE 02 INFO: QMGR EVENT SWITCH IS OFF FOR INHIBITED EVENTS
STAGE 02 INFO: QMGR EVENT SWITCH IS OFF FOR LOCAL EVENTS
STAGE 02 INFO: QMGR EVENT SWITCH IS OFF FOR PERFORMENCE EVENTS
STAGE 02 INFO: QMGR EVENT SWITCH IS OFF FOR REMOTE EVENTS
STAGE 02 INFO: QMGR EVENT SWITCH IS OFF FOR START/STOP EVENTS
STAGE 02 COMPLETE. COMPCODE : 0000 REASON CODE : 0000 SYSTEM.COMMAND.INPUT
STAGE 03 COMPLETE. COMPCODE : 0000 REASON CODE : 0000
STAGE 04 COMPLETE. COMPCODE : 0000 REASON CODE : 0000 CSQ4IVP1.BAB9810EFEAC8980
STAGE 05 COMPLETE. COMPCODE : 0000 REASON CODE : 0000
STAGE 06 COMPLETE. COMPCODE : 0000 REASON CODE : 0000
STAGE 07 COMPLETE. COMPCODE : 0000 REASON CODE : 0000
STAGE 08 COMPLETE. COMPCODE : 0000 REASON CODE : 0000 CSQ4IVP1.BAB9810F0070E645
STAGE 09 COMPLETE. COMPCODE : 0000 REASON CODE : 0000 CSQ4IVP1.BAB9812BA8706803
STAGE 10 COMPLETE. COMPCODE : 0000 REASON CODE : 0000
>>>>>>>>>>   END OF REPORT   <<<<<<<<<<
```

Figure 178. Sample report from CSQ4IVP1

Testing for queue-sharing groups

z/OS

The basic installation verification program CSQ4IVP1 tests non-shared queues.

CSQ4IVP1 can be used whether the queue manager is a member of a queue-sharing group or not. After running the basic IVP, you can test for shared queues by using the CSQ4IVP1 installation verification program with different queues. Also this tests that Db2 and the coupling facility are set up correctly.

Preparing to run CSQ4IVP1 for a queue-sharing group

Before you run CSQ4IVP1:

1. Add the coupling facility structure that the IVP uses to your CFRM policy data set, as described in “Task 10: Set up the coupling facility” on page 1476. 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 179. 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 179. 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 1524 to check the results of the IVP for queue-sharing groups.

Testing for distributed queuing

z/OS

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 180 on page 1529. 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 180 on page 1529.

```
RDEFINE MQCMDS CSQ1.DISPLAY.DQM
PERMIT CSQ1.DISPLAY.DQM CLASS(MQCMDS) ID(TS101) ACCESS(READ)

RDEFINE MQCMDS CSQ1.START.CHINIT
PERMIT CSQ1.START.CHINIT CLASS(MQCMDS) ID(TS101) ACCESS(CONTROL)

RDEFINE MQCMDS CSQ1.STOP.CHINIT
PERMIT CSQ1.STOP.CHINIT CLASS(MQCMDS) ID(TS101) ACCESS(CONTROL)

RDEFINE MQQUEUE CSQ1.SYSTEM.COMMAND.INPUT
PERMIT CSQ1.SYSTEM.COMMAND.INPUT CLASS(MQQUEUE) ID(TS101,CSQ1MSTR) ACCESS(UPDATE)

RDEFINE MQQUEUE CSQ1.SYSTEM.CSQUTIL.*
PERMIT CSQ1.SYSTEM.CSQUTIL.* CLASS(MQQUEUE) ID(TS101) ACCESS(UPDATE)

RDEFINE MQCONN CSQ1.BATCH
PERMIT CSQ1.BATCH CLASS(MQCONN) ID(TS101) ACCESS(READ)

RDEFINE MQCONN CSQ1.CHIN
PERMIT CSQ1.CHIN CLASS(MQCONN) ID(CSQ1MSTR) ACCESS(READ)

RDEFINE MQQUEUE CSQ1.SYSTEM.CHANNEL.SYNCQ
PERMIT CSQ1.SYSTEM.CHANNEL.SYNCQ CLASS(MQQUEUE) ID(CSQ1MSTR) ACCESS(UPDATE)

RDEFINE MQQUEUE CSQ1.SYSTEM.CLUSTER.COMMAND.QUEUE
PERMIT CSQ1.SYSTEM.CLUSTER.COMMAND.QUEUE CLASS(MQQUEUE) ID(CSQ1MSTR) ACCESS(ALTER)

RDEFINE MQQUEUE CSQ1.SYSTEM.CLUSTER.TRANSMIT.QUEUE
PERMIT CSQ1.SYSTEM.CLUSTER.TRANSMIT.QUEUE CLASS(MQQUEUE) ID(CSQ1MSTR) ACCESS(UPDATE)

RDEFINE MQQUEUE CSQ1.SYSTEM.CLUSTER.REPOSITORY.QUEUE
PERMIT CSQ1.SYSTEM.CLUSTER.REPOSITORY.QUEUE CLASS(MQQUEUE) ID(CSQ1MSTR) ACCESS(ALTER)

RDEFINE MQQUEUE CSQ1.SYSTEM.CHANNEL.INITQ
PERMIT CSQ1.SYSTEM.CHANNEL.INITQ CLASS(MQQUEUE) ID(CSQ1MSTR) ACCESS(UPDATE)

RDEFINE MQADMIN CSQ1.CONTEXT.**
PERMIT CSQ1.CONTEXT.** CLASS(MQADMIN) ID(CSQ1MSTR) ACCESS(CONTROL)
```

Figure 180. 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 181 on page 1530, although details might differ depending on your queue manager attributes.

- You should see the commands **(1)** each followed by several messages.
- The last message from each command should be “CSQ9022I ... NORMAL COMPLETION” **(2)**.
- The job as a whole should complete with return code zero **(3)**.

```
CSQU000I CSQUTIL IBM MQ for z/OS - V6
CSQU001I CSQUTIL Queue Manager Utility - 2005-05-09 09:06:48
COMMAND
CSQU127I CSQUTIL Executing COMMAND using input from CSQUCMD data set
CSQU120I CSQUTIL Connecting to queue manager CSQ1
CSQU121I CSQUTIL Connected to queue manager CSQ1
CSQU055I CSQUTIL Target queue manager is CSQ1
START CHINIT
(1)
CSQN205I  COUNT=    2, RETURN=00000000, REASON=00000004
CSQM138I +CSQ1 CSQMSCHI CHANNEL INITIATOR STARTING
CSQN205I  COUNT=    2, RETURN=00000000, REASON=00000000
CSQ9022I +CSQ1 CSQXCRPS ' START CHINIT' NORMAL COMPLETION
(2)
DISPLAY CHINIT
(1)
CSQN205I  COUNT=    2, RETURN=00000000, REASON=00000004
CSQM137I +CSQ1 CSQMDDQM DISPLAY CHINIT COMMAND ACCEPTED
CSQN205I  COUNT=   12, RETURN=00000000, REASON=00000000
CSQX830I +CSQ1 CSQXRDQM Channel initiator active
CSQX002I +CSQ1 CSQXRDQM Queue-sharing group is QSG1
CSQX831I +CSQ1 CSQXRDQM 8 adapter subtasks started, 8 requested
CSQX832I +CSQ1 CSQXRDQM 5 dispatchers started, 5 requested
CSQX833I +CSQ1 CSQXRDQM 0 SSL server subtasks started, 0 requested
CSQX840I +CSQ1 CSQXRDQM 0 channel connections current, maximum 200
CSQX841I +CSQ1 CSQXRDQM 0 channel connections active, maximum 200,
including 0 paused
CSQX842I +CSQ1 CSQXRDQM 0 channel connections starting,
0 stopped, 0 retrying
CSQX836I +CSQ1 Maximum channels - TCP/IP 200, LU 6.2 200
CSQX845I +CSQ1 CSQXRDQM TCP/IP system name is TCPIP
CSQX848I +CSQ1 CSQXRDQM TCP/IP listener INDISP=QMGR not started
CSQX848I +CSQ1 CSQXRDQM TCP/IP listener INDISP=GROUP not started
CSQX849I +CSQ1 CSQXRDQM LU 6.2 listener INDISP=QMGR not started
CSQX849I +CSQ1 CSQXRDQM LU 6.2 listener INDISP=GROUP not started
CSQ9022I +CSQ1 CSQXCRPS ' DISPLAY CHINIT' NORMAL COMPLETION
(2)
STOP CHINIT
(1)
CSQN205I  COUNT=    2, RETURN=00000000, REASON=00000004
CSQM137I +CSQ1 CSQMTCHI STOP CHINIT COMMAND ACCEPTED
CSQN205I  COUNT=    2, RETURN=00000000, REASON=00000000
CSQ9022I +CSQ1 CSQXCRPS ' STOP CHINIT' NORMAL COMPLETION
(2)
CSQU057I CSQUCMDS 3 commands read
CSQU058I CSQUCMDS 3 commands issued and responses received, 0 failed
CSQU143I CSQUTIL 1 COMMAND statements attempted
CSQU144I CSQUTIL 1 COMMAND statements executed successfully
CSQU148I CSQUTIL Utility completed, return code=0
(3)
```

Figure 181. Example output from CSQ4IVPX

Testing for C, C++, COBOL, PL/I, and CICS programs with IBM MQ for z/OS

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 Sample programs for IBM MQ for z/OS.

Setting up communications with other queue managers

This section describes the IBM MQ for z/OS preparations you need to make before you can start to use distributed queuing.

To define your distributed-queuing requirements, you need to define the following items:
- Define the channel initiator procedures and data sets
- Define the channel definitions
- Define the queues and other objects
- Define access security

To enable distributed queuing, you must perform the following three tasks:
- Customize the distributed queuing facility and define the IBM MQ objects required as described in Defining system objects and "Preparing to customize queue managers on z/OS" on page 1455.
- 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 1553.

If you are using queue-sharing groups, see Distributed queuing and queue-sharing groups.

See the following sections for additional considerations for using distributed queuing with IBM MQ for z/OS.

Operator messages

Because the channel initiator uses a number of asynchronously operating dispatchers, operator messages might occur on the log out of chronological sequence.

Channel operation commands

Channel operation commands generally involve two stages. When the command syntax has been checked and the existence of the channel verified, a request is sent to the channel initiator. Message CSQM134I or CSQM137I is sent to the command issuer to indicate the completion of the first stage. When the channel initiator has processed the command, further messages indicating its success or otherwise are sent to the command issuer along with message CSQ9022I or CSQ9023I. Any error messages generated could also be sent to the z/OS console.

All cluster commands except DISPLAY CLUSQMGR, however, work asynchronously. Commands that change object attributes update the object and send a request to the channel initiator. Commands for working with clusters are checked for syntax and a request is sent to the channel initiator. In both cases, message CSQM130I is sent to the command issuer indicating that a request has been sent. This message is followed by message CSQ9022I to indicate that the command has completed successfully, in that a request has been sent. It does not indicate that the cluster request has completed successfully. The requests sent to the channel initiator are processed asynchronously, along with cluster requests received from other members of the cluster. In some cases, these requests must be sent to the whole cluster to determine if they are successful or not. Any errors are reported to the z/OS on the system where the channel initiator is running. They are not sent to the command issuer.

Undelivered-message queue

A Dead Letter handler is provided with IBM MQ for z/OS. See The dead-letter queue handler utility (CSQUDLQH) for more information.

Queues in use

MCAs for receiver channels can keep the destination queues open even when messages are not being transmitted. This behavior results in the queues appearing to be 'in use'.

Security changes

If you change security access for a user ID, the change might not take effect immediately. (See one of Security considerations for the channel initiator on z/OS , Profiles for queue security, and “Task 11: Implement your ESM security controls” on page 1478 for more information.)

Communications stopped - TCP

If TCP is stopped for some reason and then restarted, the IBM MQ for z/OS TCP listener waiting on a TCP port is stopped.

Automatic channel-reconnect allows the channel initiator to detect that TCP/IP is unavailable and to automatically restart the TCP/IP listener when TCP/IP returns. This automatic restart alleviates the need for operations staff to notice the problem with TCP/IP and manually restart the listener. While the listener is out of action, the channel initiator can also be used to try the listener again at the interval specified by LSTRTMR in the channel initiator parameter module. These attempts can continue until TCP/IP returns and the listener successfully restarts automatically. For information about LSTRTMR, see Security concepts on z/OS and Distributed queuing messages (CSQX...).

Communications stopped - LU6.2

If APPC is stopped, the listener is also stopped. Again, in this case, the listener automatically tries again at the LSTRTMR interval so that, if APPC restarts, the listener can restart too.

If the Db2 fails, shared channels that are already running continue to run, but any new channel start requests fail. When the Db2 is restored new requests are able to complete.

z/OS Automatic Restart Management (ARM)

Automatic restart management (ARM) is a z/OS recovery function that can improve the availability of specific batch jobs or started tasks (for example, subsystems). It can therefore result in a faster resumption of productive work.

To use ARM, you must set up your queue managers and channel initiators in a particular way to make them restart automatically. For information, see Using the z/OS Automatic Restart Manager (ARM).

Related concepts:

"Customizing IBM MQ for z/OS" on page 1459
Use this topic as a step by step guide for customizing your IBM MQ system.

"Monitoring and controlling channels on z/OS" on page 1534
Use the DQM commands and panels to create, monitor, and control the channels to remote queue managers.

"Setting up communication for z/OS" on page 1553
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 1558
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 1563
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 961
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

Use one of the IBM MQ command input methods to define IBM MQ objects. Refer to the information within this topic for further details about defining these objects.

Refer to "Monitoring and controlling channels on z/OS" on page 1534 for information about defining objects.

Transmission queues and triggering channels

Define the following:

- A local queue with the usage of XMITQ for each sending message channel.
- Remote queue definitions.

  A remote queue object has three distinct uses, depending upon the way the name and content are specified:
  - Remote queue definition
  - Queue manager alias definition
  - Reply-to queue alias definition

  These three ways are shown in Three ways of using the remote queue definition object.

Use the TRIGDATA field on the transmission queue to trigger the specified channel. For example:

```
DEFINE QLOCAL(MYXMITQ) USAGE(XMITQ) TRIGGER +
INITQ(SYSTEM.CHANNEL.INITQ) TRIGDATA(MYCHANNEL)
DEFINE CHL(MYCHANNEL) CHLTYPE(SDR) TRPTYPE(TCP) +
XMITQ(MYXMITQ) CONNAME('9.20.9.30(1555)')
```

The supplied sample CSQ4INYD gives additional examples of the necessary definitions.

z/OS Loss of connectivity to the CF structure where the synchronization queue for shared channels is defined, or similar problems, might temporarily prevent a channel from starting. After problem resolution, if you are using a trigger type of FIRST and the channel fails to start when it is triggered, you must start the channel manually. If you want to automatically start triggered channels after problem resolution, consider setting the queue manager TRIGINT attribute to a value other than the default. Setting the TRIGINT attribute to a value other than the default causes the channel initiator to retry starting the channel periodically while there are messages on the transmission queue.

Synchronization queue

DQM requires a queue for use with sequence numbers and logical units of work identifiers (LUWID). You must ensure that a queue is available with the name SYSTEM.CHANNEL.SYNCQ (see Planning on z/OS). This queue must be available otherwise the channel initiator cannot start.

Make sure that you define this queue using INDXTYPE(MSGID). This attribute improves the speed at which they can be accessed.

Channel command queues

You need to ensure that a channel command queue exists for your system with the name SYSTEM.CHANNEL.INITQ.

If the channel initiator detects a problem with the SYSTEM.CHANNEL.INITQ, it is unable to continue normally until the problem is corrected. The problem could be one of the following:
- The queue is full
- The queue is not enabled for put
- The page set that the queue is on is full
- The channel initiator does not have the correct security authorization to the queue

If the definition of the queue is changed to GET(DISABLED) while the channel initiator is running, the initiator is unable to get messages from the queue, and terminates.

Starting the channel initiator

Triggering is implemented using the channel initiator. On IBM MQ for z/OS, the initiator is started with the MQSC command `START CHINIT`.

Stopping the channel initiator

The channel initiator is stopped automatically when you stop the queue manager. If you need to stop the channel initiator but not the queue manager, you can use the MQSC command `STOP CHINIT`.

Monitoring and controlling channels on z/OS

Use the DQM commands and panels to create, monitor, and control the channels to remote queue managers.

Each z/OS queue manager has a DQM program (the *channel initiator*) for controlling interconnections to remote queue managers using native z/OS facilities.

The implementation of these panels and commands on z/OS is integrated into the operations and control panels and the MQSC commands. No differentiation is made in the organization of these two sets of panels and commands.

You can also enter commands using Programmable Command Format (PCF) commands. See Automating administration tasks for information about using these commands.

The information in this section applies in all cases where the channel initiator is used for distributed queuing. It applies whether you are using queue-sharing groups, or intra-group queuing.

The DQM channel control function

For an overview of the distributed queue management model, see “Message sending and receiving” on page 983.

The channel control function consists of panels, commands and programs, two synchronization queues, channel command queues, and the channel definitions. This topic is a brief description of the components of the channel control function.

- The channel definitions are held as objects in page set zero or in Db2, like other IBM MQ objects in z/OS.
- You use the operations and control panels, MQSC commands, or PCF commands to:
  - Create, copy, display, alter, and delete channel definitions
  - Start and stop channel initiators and listeners
  - Start, stop, and ping channels, reset channel sequence numbers, and resolve in-doubt messages when links cannot be re-established
  - Display status information about channels
  - Display information about DQM

  In particular, you can use the CSQINPX initialization input data set to issue your MQSC commands. This set can be processed every time you start the channel initiator. For more information, see Initialization commands.
- There are two queues (SYSTEM.CHANNEL.SYNCQ and SYSTEM.QSG.CHANNEL.SYNCQ) used for channel re-synchronization purposes. Define these queues with INDXTYPE(MSGID) for performance reasons.
- The channel command queue (SYSTEM.CHANNEL.INITQ) is used to hold commands for channel initiators, channels, and listeners.
- The channel control function program runs in its own address space, separate from the queue manager, and comprises the channel initiator, listeners, MCAs, trigger monitor, and command handler.
- For queue-sharing groups and shared channels, see Shared queues and queue-sharing groups.
- For intra-group queuing, see Intra-group queuing

Managing your channels on z/OS

Use the links in the following table for information about how to manage your channels, channel initiators, and listeners:

Table 149. Channel tasks

Task to be performed	MQSC command
Define a channel	DEFINE CHANNEL
Alter a channel definition	ALTER CHANNEL
Display a channel definition	DISPLAY CHANNEL
Delete a channel definition	DELETE CHANNEL
Start a channel initiator	START CHINIT
Stop a channel initiator	STOP CHINIT
Display channel initiator information	DISPLAY CHINIT
Start a channel listener	START LISTENER
Stop a channel listener	STOP LISTENER
Start a channel	START CHANNEL
Test a channel	PING CHANNEL

Table 149. Channel tasks (continued)

Task to be performed	MQSC command
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"
You can use the MQSC commands, the PCF commands, or the operations and control panels to manage DQM.

"Setting up communications with other queue managers" on page 1531
This section describes the IBM MQ for z/OS preparations you need to make before you can start to use distributed queuing.

"Customizing IBM MQ for z/OS" on page 1459
Use this topic as a step by step guide for customizing your IBM MQ system.

"Setting up communication for z/OS" on page 1553
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 1558
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 1563
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 961
This section provides more detailed information about intercommunication between IBM MQ installations, including queue definition, channel definition, triggering, and sync point procedures

Using the panels and the commands:

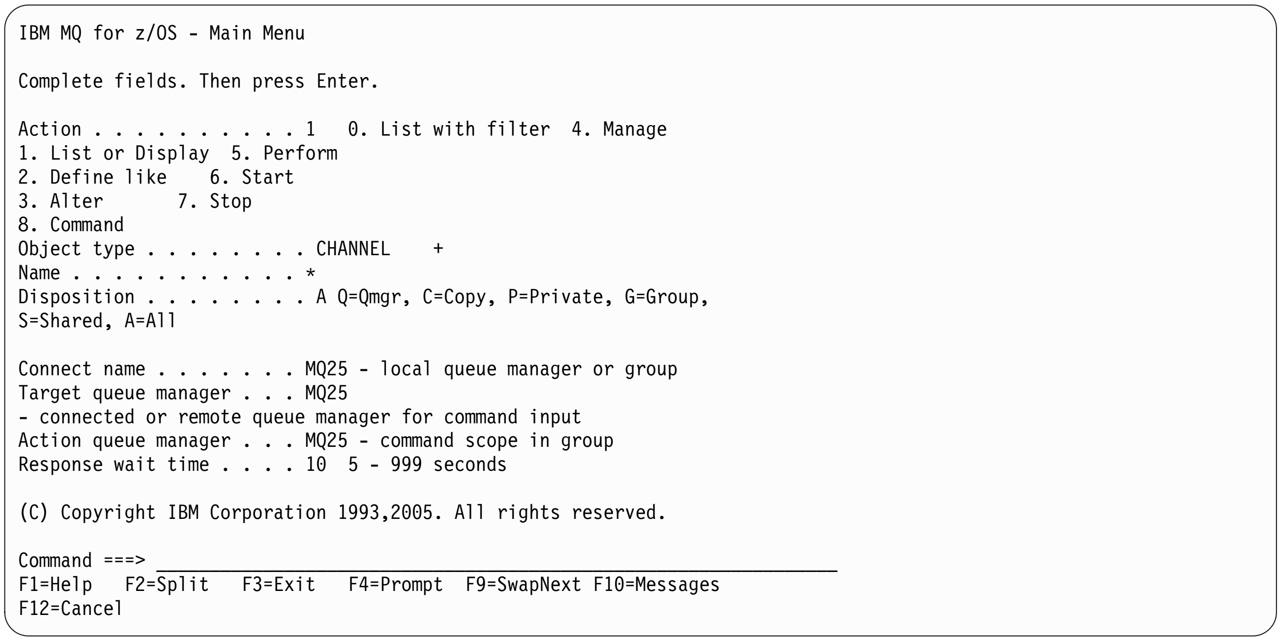
You can use the MQSC commands, the PCF commands, or the operations and control panels to manage DQM.

For information about the syntax of the MQSC commands, see Script (MQSC) Commands. For information about PCF commands, see Introduction to Programmable Command Formats.

Using the initial panel

For an introduction to invoking the operations and control panels, using the function keys, and getting help, see Administering IBM MQ for z/OS.

Note: To use the operations and control panels, you must have the correct security authorization; see Administering IBM MQ for z/OS and sub topics for more information. Figure 182 on page 1537 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.

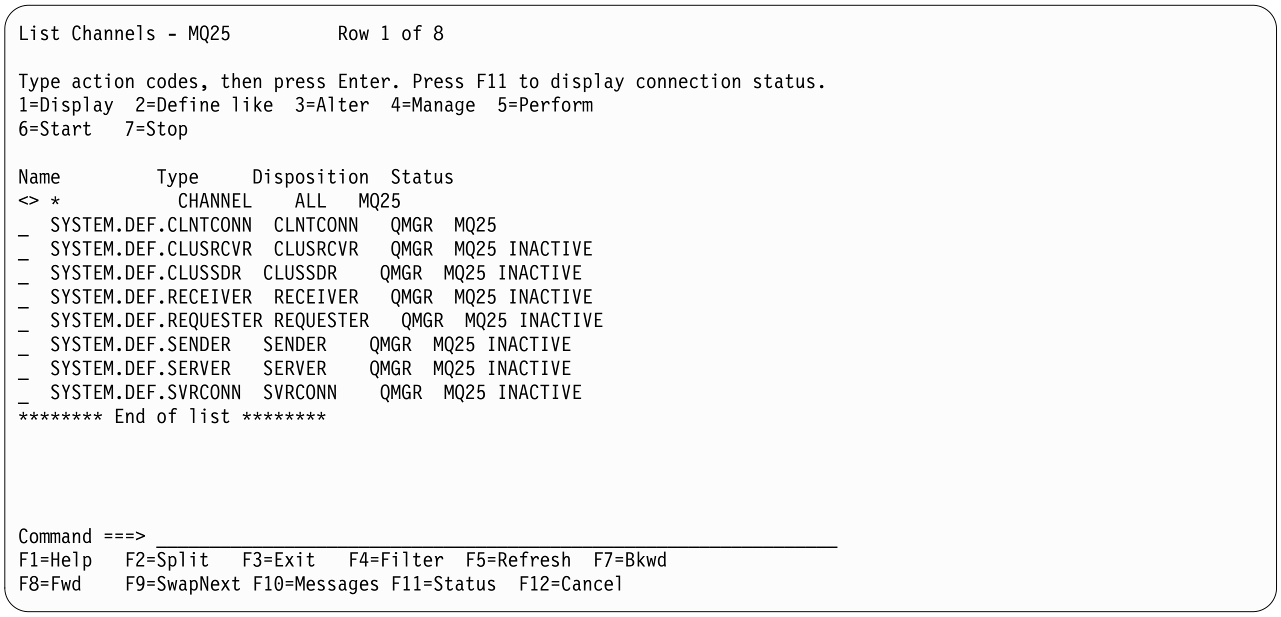


```
IBM MQ for z/OS - Main Menu

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 1993,2005. All rights reserved.

Command ===> ______________________________________________________________
F1=Help   F2=Split   F3=Exit   F4=Prompt  F9=SwapNext F10=Messages
F12=Cancel
```

Figure 182. 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 183 on page 1538 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 183. Listing channels

Defining a channel:

You can define a channel using MQSC commands or using the operations and control panels.

To define a channel using the MQSC commands, use DEFINE CHANNEL.

Using the operations and control panels, starting from the initial panel, complete these fields and press enter:

Field	**Value**
Action	2 (Define like)
Object type	channel type (for example SENDER) or CHANNEL
Name	
Disposition	The location of the new object.

You are presented with some panels to complete with information about the name and attributes you want for the channel you are defining. They are initialized with the default attribute values. Change any you want before pressing enter.

Note: If you entered CHANNEL in the **`object type`** field, you are presented with the Select a Valid Channel Type panel first.

If you want to define a channel with the same attributes as an existing channel, put the name of the channel you want to copy in the **`Name`** field on the initial panel. The panels are initialized with the attributes of the existing object.

For information about the channel attributes, see Channel attributes

Note:

1. Name all the channels in your network uniquely. As shown in Network diagram showing all channels, including the source and target queue manager names in the channel name is a good way to do this naming.

After you have defined your channel you must secure your channel, see "Securing a channel" on page 1541

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:

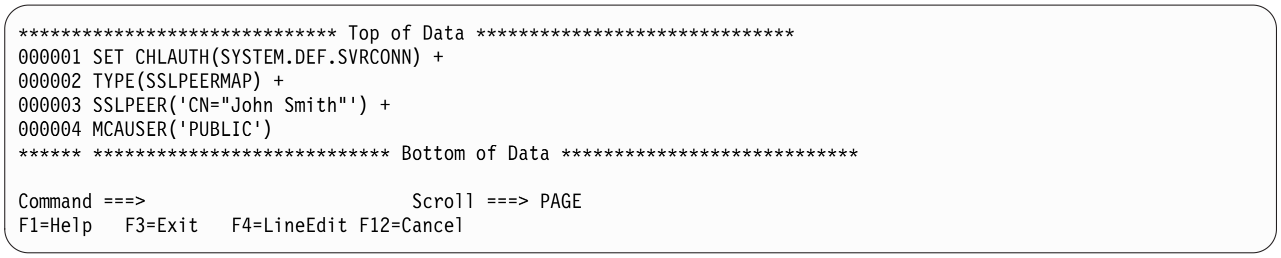
You can secure a channel using MQSC commands or using the operations and control panels.

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

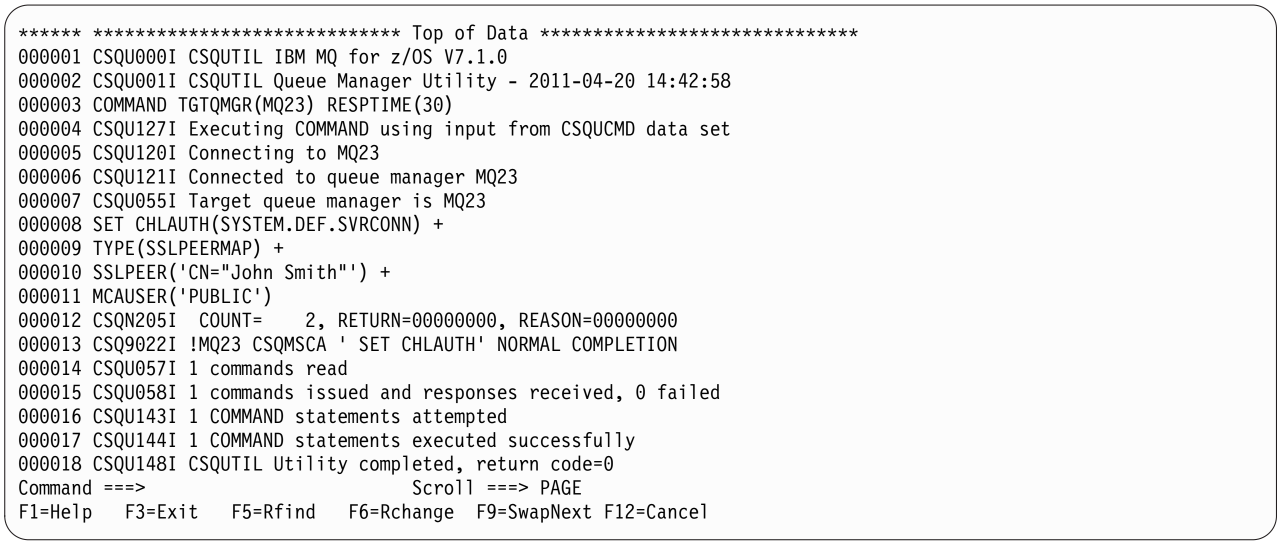
You are presented with an editor within which you can provide an MQSC command, in this case a CHLAUTH command, see Figure 184. When you have finished typing in the command, the plus signs (+) are needed. Type PF3 to exit from the editor and submit the command to the command server.



```
****************************** 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 184. Command Entry

The output of the command is then presented to you, see Figure 185



```
****** ****************************** Top of Data ******************************
000001 CSQU000I CSQUTIL IBM MQ for z/OS V7.1.0
000002 CSQU001I CSQUTIL Queue Manager Utility - 2011-04-20 14:42:58
000003 COMMAND TGTQMGR(MQ23) RESPTIME(30)
000004 CSQU127I Executing COMMAND using input from CSQUCMD data set
000005 CSQU120I Connecting to MQ23
000006 CSQU121I Connected to queue manager MQ23
000007 CSQU055I Target queue manager is MQ23
000008 SET CHLAUTH(SYSTEM.DEF.SVRCONN) +
000009 TYPE(SSLPEERMAP) +
000010 SSLPEER('CN="John Smith"') +
000011 MCAUSER('PUBLIC')
000012 CSQN205I  COUNT=    2, RETURN=00000000, REASON=00000000
000013 CSQ9022I !MQ23 CSQMSCA ' SET CHLAUTH' NORMAL COMPLETION
000014 CSQU057I 1 commands read
000015 CSQU058I 1 commands issued and responses received, 0 failed
000016 CSQU143I 1 COMMAND statements attempted
000017 CSQU144I 1 COMMAND statements executed successfully
000018 CSQU148I CSQUTIL Utility completed, return code=0
Command ===>                         Scroll ===> PAGE
F1=Help   F3=Exit   F5=Rfind   F6=Rchange  F9=SwapNext F12=Cancel
```

Figure 185. Command Output

Starting a channel initiator:

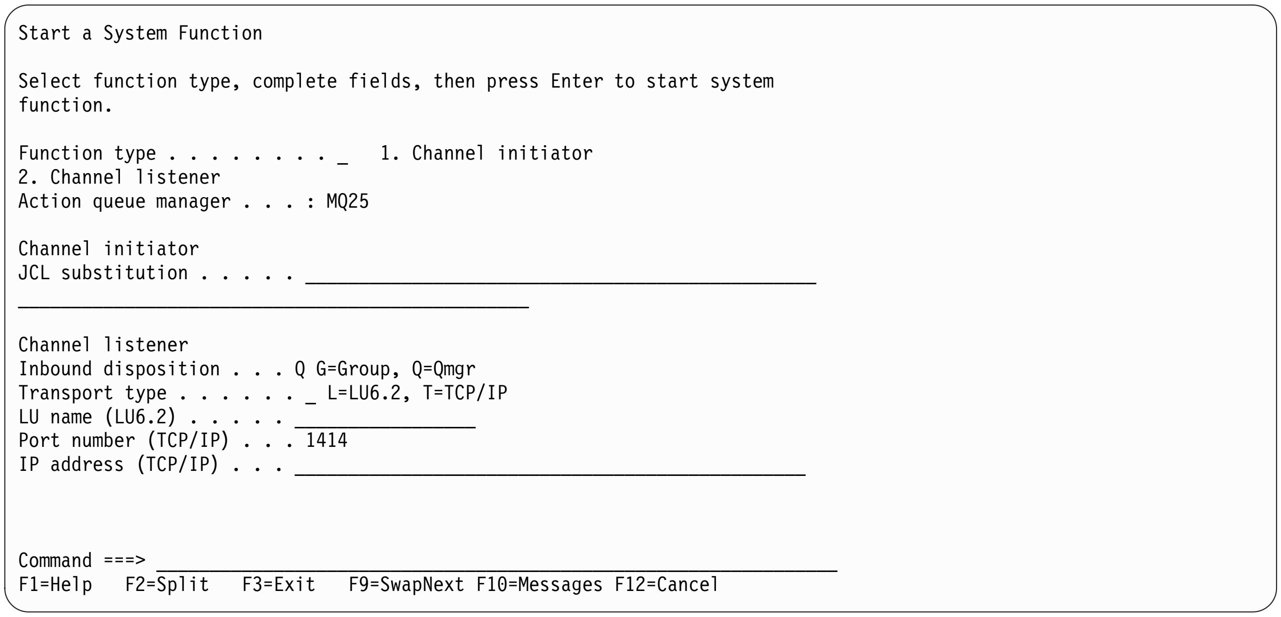
You can start a channel initiator using MQSC commands or using the operations and control panels.

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 186. Starting a system function

Select function type 1 (channel initiator), and press enter.

Stopping a channel initiator:

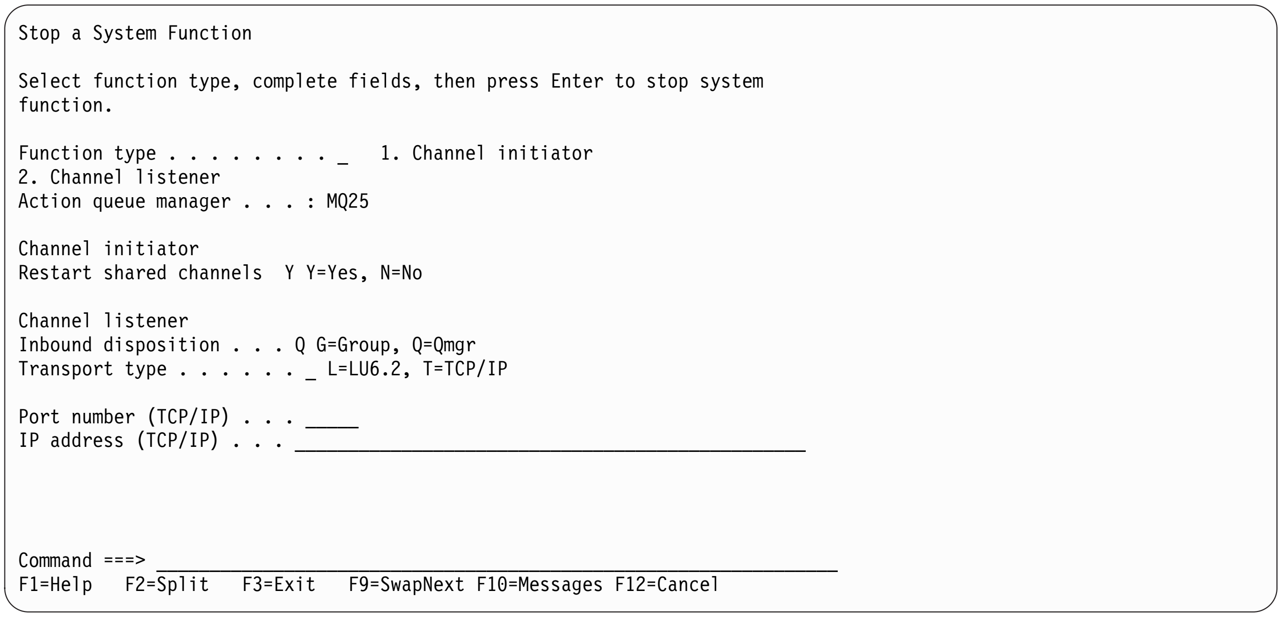
You can stop a channel initiator using MQSC commands or using the operations and control panels.

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 187. 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 186 on page 1542).

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 187 on page 1543).

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:

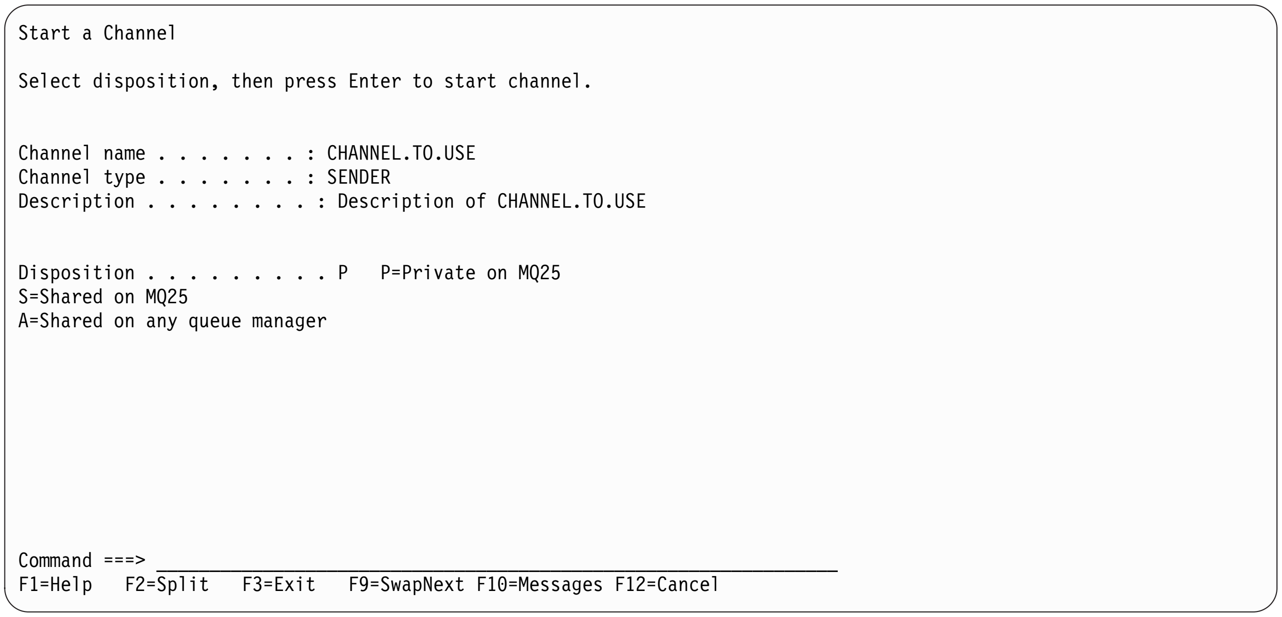
You can start a channel using MQSC commands or using the operations and control panels.

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 188. 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:

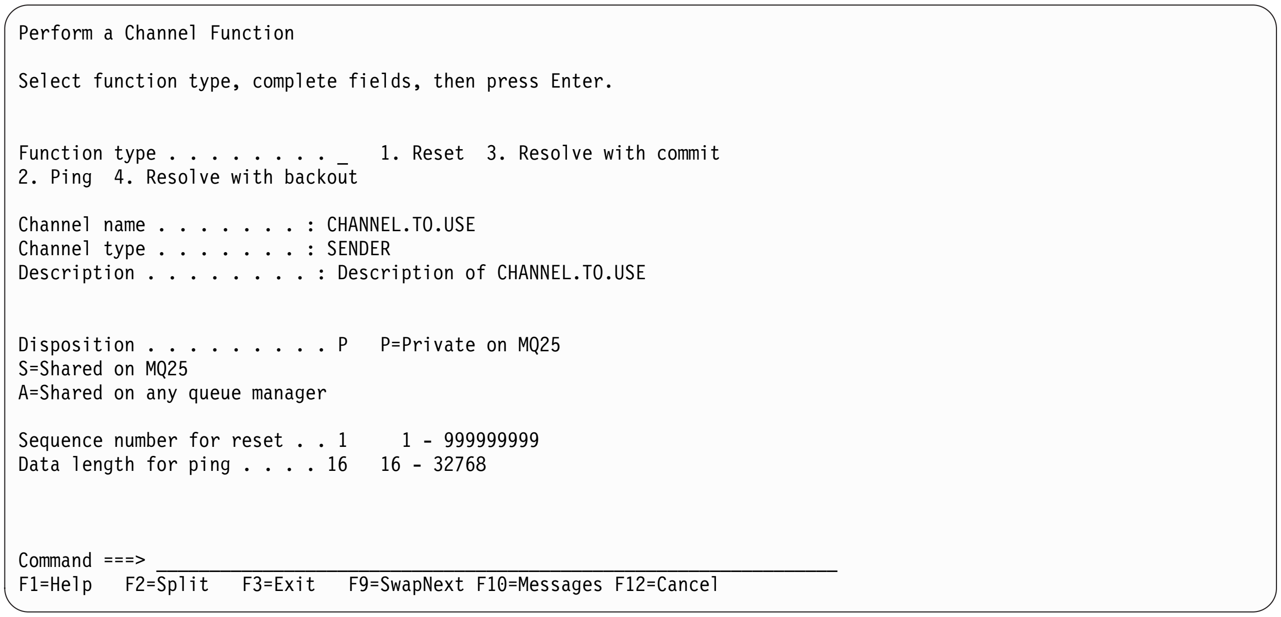
You can test a channel using MQSC commands or using the operations and control panels.

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 189. 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 189).

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 189 on page 1547).

Select Function type 3 or 4 (resolve with commit or backout). (See “In-doubt channels” on page 1004 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:

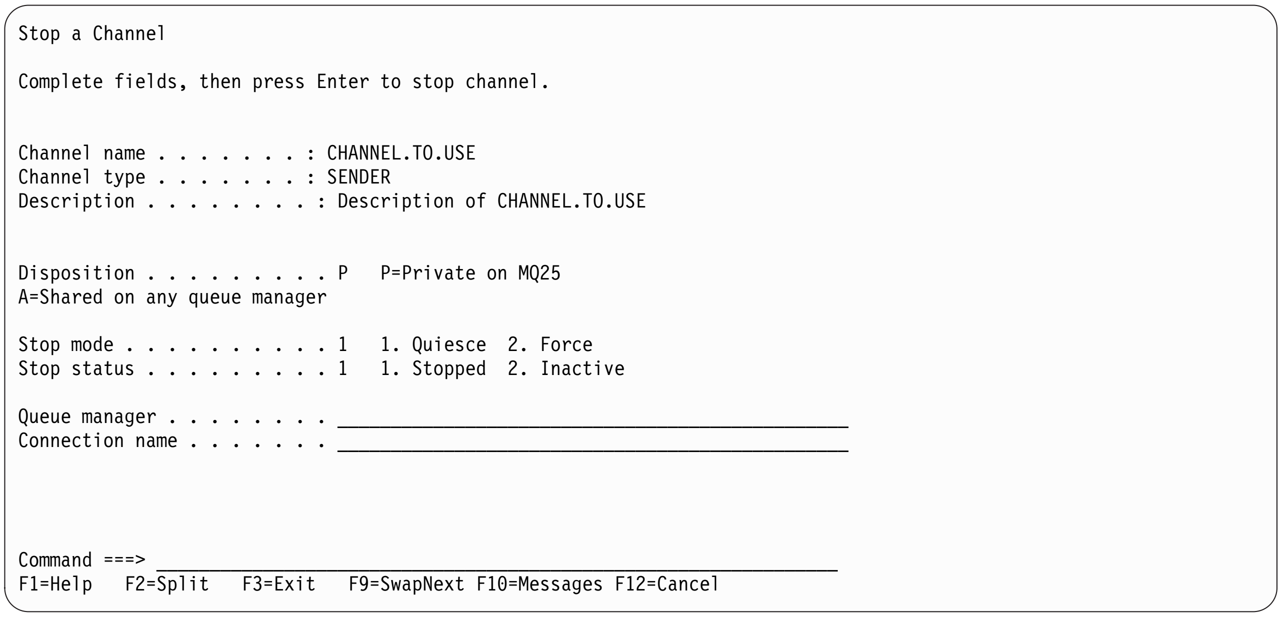
You can stop a channel using MQSC commands or using the operations and control panels.

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 190. 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 1002 for more information. For information about restarting stopped channels, see “Restarting stopped channels” on page 1003.

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 183 on page 1538), a summary of the channel status is shown for each channel as follows:

INACTIVE	No connections are active
status	One connection is active
nnn status	More than one connection is current and all current connections have the same status
nnn CURRENT	More than one connection is current and the current connections do not all have the same status
Blank	IBM MQ is unable to determine how many connections are active (for example, because the channel initiator is not running) **Note:** For channel objects with the disposition GROUP, no status is displayed.

where *nnn* is the number of active connections, and *status* is one of the following:

INIT	INITIALIZING
BIND	BINDING
START	STARTING
RUN	RUNNING
STOP	STOPPING or STOPPED
RETRY	RETRYING
REQST	REQUESTING

To display more information about the channel status, press the Status key (F11) on the List Channel or the Display, or Alter channel panels to display the List Channels - Current Status panel (see Figure 191 on page 1551).

```
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 191. 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 993 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 192, in which the information for each cluster channel occupies three lines, and includes its channel, cluster, and queue manager names. For cluster-sender channels, the overall state is shown.

```
List Cluster queue manager Channels - MQ25    Row 1 of 9

Type action codes, then press Enter. Press F11 to display connection status.
1=Display  5=Perform  6=Start  7=Stop

Channel name     Connection name                 State
Type     Cluster name                   Suspended
Cluster queue manager name            Disposition
<>   *                        -    MQ25
_ TO.MQ90.T      HURSLEY.MACH90.COM(1590)
_  CLUSRCVR   VJH01T                     N
_   MQ90                       -    MQ25
_ TO.MQ95.T      HURSLEY.MACH95.COM(1595)            RUN
_  CLUSSDRA   VJH01T                     N
_   MQ95                       -    MQ25
_ TO.MQ96.T      HURSLEY.MACH96.COM(1596)            RUN
_  CLUSSDRB   VJH01T                     N
_   MQ96                       -    MQ25
******** End of list ********


Command ===> ________________________________________________________________
F1=Help   F2=Split   F3=Exit   F4=Filter  F5=Refresh  F7=Bkwd
F8=Fwd    F9=SwapNext F10=Messages F11=Status  F12=Cancel
```

Figure 192. 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, using a minimum z/OS level of z/OS Version 2.1.

See IBM zEnterprise Data Compression (zEDC) for z/OS for further information.

To start using zEDC for channel compression, you must:
- Configure the zEDC Software License in the IFAPRDxx parmlib member.
- 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.
- 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 data to be compressed is above a size defined by the zEDC Express facility, currently 4 KB. For messages below the threshold size, compression is performed in the software.

Setting up communication for z/OS

When a distributed-queuing management channel is started, it tries to use the connection specified in the channel definition. To succeed, it is necessary for the connection to be defined and available. This section explains how to define a connection.

DQM is a remote queuing facility for IBM MQ. It provides channel control programs for the queue manager that form the interface to communication links. These links are controllable by the system operator. The channel definitions held by distributed queuing management use these connections.

Choose from one of the two forms of communication protocol that can be used for z/OS:
- "Defining a TCP connection on z/OS" on page 1554
- "Defining an LU6.2 connection for z/OS using APPC/MVS" on page 1557

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 1563.

Related concepts:

"Using the panels and the commands" on page 1536
You can use the MQSC commands, the PCF commands, or the operations and control panels to manage DQM.

"Setting up communications with other queue managers" on page 1531
This section describes the IBM MQ for z/OS preparations you need to make before you can start to use distributed queuing.

"Customizing IBM MQ for z/OS" on page 1459
Use this topic as a step by step guide for customizing your IBM MQ system.

"Monitoring and controlling channels on z/OS" on page 1534
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 1558
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 1563
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 961
This section provides more detailed information about intercommunication between IBM MQ installations, including queue definition, channel definition, triggering, and sync point procedures

Defining a TCP connection on z/OS:

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

The TCP address space name must be specified in the TCP system parameters data set, *tcpip*.TCPIP.DATA. In the data set, a "TCPIPJOBNAME *TCPIP_proc*" statement must be included.

If you are using a firewall, you need to configure `allow` connections from the channel initiator to the addresses in the channels, and from the remote connections into the queue manager.

Typically the definition for a firewall configures the sending IP address and port to the destination IP address and port:

- A z/OS image can have more than one host name, and you might need to configure the firewall with multiple host addresses as the source address.

  You can use the NETSTAT HOME command to display these names and addresses.
- A channel initiator can have multiple listeners on different ports, so you need to configure these ports.
- If you are using a shared port for a queue-sharing group you must configure the shared port as well.

The channel initiator address space must have authority to read the data set. The following techniques can be used to access your TCPIP.DATA data set, depending on which TCP/IP product and interface you are using:

- Environment variable, RESOLVER_CONFIG
- HFS file, /etc/resolv.conf
- //SYSTCPD DD statement
- //SYSTCPDD DD statement
- *jobname/userid*.TCPIP.DATA
- SYS1.TCPPARMS(TCPDATA)

- *zapname*.TCPIP.DATA

You must also be careful to specify the high-level qualifier for TCP/IP correctly.

You need a suitably configured Domain Name System (DNS) server, capable of both Name to IP address translation and IP address to Name translation.

Note: Some changes to the resolver configuration require a recycle of applications using it, for example, IBM MQ.

For more information, see the following:
- Base TCP/IP system
- z/OS UNIX System Services.

Each TCP channel when started uses TCP resources; you might need to adjust the following parameters in your PROFILE.TCPIP configuration data set:

ACBPOOLSIZE
Add one per started TCP channel, plus one

CCBPOOLSIZE
Add one per started TCP channel, plus one per DQM dispatcher, plus one

DATABUFFERPOOLSIZE
Add two per started TCP channel, plus one

MAXFILEPROC
Controls how many channels each dispatcher in the channel initiator can handle.

This parameter is specified in the BPXPRMxx member of SYSI.PARMLIB. Ensure that you specify a value large enough for your needs.

By default, the channel initiator is only capable of binding to IP addresses associated with the stack named in the TCPNAME queue manager attribute. To allow the channel initiator to communicate using additional TCP/IP stacks on the system, change the TCPSTACK queue manager attribute to MULTIPLE.

Related concepts:

"Sending end"
At the sending end of the TCP/IP connection, there are a number of settings to configure.

"Receiving on TCP" on page 1556
At the receiving end of the TCP/IP connection, there are a number of settings to configure.

"Using the TCP listener backlog option" on page 1556
When receiving on TCP/IP, a maximum number of outstanding connection requests is set. These outstanding requests can be considered a *backlog* of requests waiting on the TCP/IP port for the listener to accept the request.

Sending end:

At the sending end of the TCP/IP connection, there are a number of settings to configure.

The connection name (CONNAME) field in the channel definition must be set to either the host name (for example MVSHUR1) or the TCP network address of the target. The TCP network address can be in IPv4 dotted decimal form (for example 127.0.0.1) or IPv6 hexadecimal form (for example 2001:DB8:0:0:0:0:0:0). If the connection name is a host name, a TCP name server is required to convert the host name into a TCP host address. (This requirement is a function of TCP, not IBM MQ.)

On the initiating end of a connection (sender, requester, and server channel types) it is possible to provide an optional port number for the connection, for example:

Connection name
192.0.2.0(1555)

In this case the initiating end attempts to connect to a receiving program listening on port 1555.

Note: The default port number of 1414 is used if an optional port number is not specified.

The channel initiator can use any TCP/IP stack which is active and available. By default, the channel initiator binds its outbound channels to the default IP address for the TCP/IP stack named in the TCPNAME queue manager attribute. To connect through a different stack, you need to specify either the host name or IP address of the stack in the LOCLADDR attribute of the channel.

Receiving on TCP:

At the receiving end of the TCP/IP connection, there are a number of settings to configure.

Receiving channel programs are started in response to a startup request from the sending channel. To do so, a listener program has to be started to detect incoming network requests and start the associated channel. You start this listener program with the START LISTENER command, or using the operations and control panels.

By default:
- The TCP Listener program uses port 1414 and listens on all addresses available to your TCP stack.
- TCP/IP listeners can bind only to addresses associated with the TCP/IP stack named in the TCPNAME queue manager attribute.

To start listeners for other addresses, or all available TCP stacks, set your TCPSTACK queue manager attribute to 'MULTIPLE'.

You can start your TCP listener program to listen only on a specific address or host name by specifying IPADDR in the START LISTENER command. For more information, see Listeners.

Using the TCP listener backlog option:

When receiving on TCP/IP, a maximum number of outstanding connection requests is set. These outstanding requests can be considered a *backlog* of requests waiting on the TCP/IP port for the listener to accept the request.

The default listener backlog value on z/OS is 255. If the backlog reaches this values, the TCP/IP connection is rejected and the channel is not able to start.

For MCA channels, this results in the channel going into a RETRY state and retrying the connection at a later time.

For client connections, the client receives an MQRC_Q_MGR_NOT_AVAILABLE reason code from MQCONN and can retry the connection at a later time.

Defining an LU6.2 connection for z/OS using APPC/MVS:

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

APPC/MVS setup

Each instance of the channel initiator must have the name of the LU that it is to use defined to APPC/MVS, in the APPCPMxx member of SYS1.PARMLIB, as in the following example:

```
LUADD ACBNAME( luname ) NOSCHED TPDATA(CSQ.APPCTP)
```

`luname` is the name of the logical unit to be used. `NOSCHED` is required; `TPDATA` is not used. No additions are necessary to the ASCHPMxx member, or to the APPC/MVS TP profile data set.

The side information data set must be extended to define the connections used by DQM. See the supplied sample CSQ4SIDE for details of how to do this using the APPC utility program ATBSDFMU. For details of the TPNAME values to use, see the *Multiplatform APPC Configuration Guide* ("Redbook") and the following table for information:

Table 150. Settings on the local z/OS system for a remote queue manager platform

Remote platform	TPNAME
z/OS or MVS	The same as TPNAME in the corresponding side information about the remote queue manager.
IBM i	The same as the compare value in the routing entry on the IBM i system.
HP Integrity NonStop Server	The same as the TPNAME specified in the receiver-channel definition.
UNIX and Linux systems	The same as TPNAME in the corresponding side information about the remote queue manager.
Windows	As specified in the Windows Run Listener command, or the invokable Transaction Program that was defined using TpSetup on Windows.

If you have more than one queue manager on the same machine, ensure that the TPnames in the channel definitions are unique.

See the *Multiplatform APPC Configuration Guide* also for information about the VTAM definitions that might be required.

In an environment where the queue manager is communicating using APPC with a queue manager on the same or another z/OS system, ensure that either the VTAM definition for the communicating LU specifies SECACPT(ALREADYV), or that there is a RACF APPCLU profile for the connection between LUs, which specifies CONVSEC(ALREADYV).

The z/OS command VARY ACTIVE must be issued against both base and listener LUs before attempting to start either inbound or outbound communications.

Related concepts:

"Connecting to LU 6.2"
To connect to LU 6.2, there are a number of settings to configure.

"Receiving on LU 6.2"
To receive on LU 6.2, there are a number of settings to configure.

Connecting to LU 6.2:

To connect to LU 6.2, there are a number of settings to configure.

The connection name (CONNAME) field in the channel definition must be set to the symbolic destination name, as specified in the side information data set for APPC/MVS.

The LU name to use (defined to APPC/MVS as described previously) must also be specified in the channel initiator parameters. It must be set to the same LU that is used for receiving by the listener.

The channel initiator uses the "SECURITY(SAME)" APPC/MVS option, so it is the user ID of the channel initiator address space that is used for outbound transmissions, and is presented to the receiver.

Receiving on LU 6.2:

To receive on LU 6.2, there are a number of settings to configure.

Receiving MCAs are started in response to a startup request from the sending channel. To do so, a listener program has to be started to detect incoming network requests and start the associated channel. The listener program is an APPC/MVS server. You start it with the START LISTENER command, or using the operations and control panels. You must specify the LU name to use with a symbolic destination name defined in the side information data set. The local LU so identified must be the same as the one used for outbound transmissions, as set in the channel initiator parameters.

Preparing IBM MQ for z/OS for DQM with queue-sharing groups

Use the instructions in this section to configure distributed queuing with queue-sharing groups on IBM MQ for z/OS.

For an example configuration using queue-sharing groups, see Example configuration - IBM MQ for z/OS using queue-sharing groups. For a message channel planning example using queue-sharing groups, see Message channel planning example for z/OS using queue-sharing groups.

You need to create and configure the following components to enable distributed queuing with queue-sharing groups:

- LU 6.2 and TCP/IP listeners
- Transmission queues and triggering
- Message channel agents
- Synchronization queue

After you have created the components you need to set up the communication, see "Setting up communication for IBM MQ for z/OS using queue-sharing groups" on page 1563.

For information about how to monitor and control channels when using queue-sharing groups, see "Monitoring and controlling channels on z/OS" on page 1534.

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 1560.

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 1560.

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:

"Customizing IBM MQ for z/OS" on page 1459
Use this topic as a step by step guide for customizing your IBM MQ system.

"Setting up communications with other queue managers" on page 1531
This section describes the IBM MQ for z/OS preparations you need to make before you can start to use distributed queuing.

"Clusters and queue-sharing groups" on page 1562
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 1562
During shared queue peer recovery, message channel agents that process messages on shared queues serialize their access to the queues.

Related tasks:

"Configuring distributed queuing" on page 961
This section provides more detailed information about intercommunication between IBM MQ installations, including queue definition, channel definition, triggering, and sync point procedures

Related information:

Shared queues and queue-sharing groups

Intra-group queuing

LU 6.2 and TCP/IP listeners for queue-sharing groups:

The group LU 6.2 and TCP/IP listeners listen on an address that is logically connected to the generic address.

For the LU 6.2 listener, the specified LUGROUP is mapped to the VTAM generic resource associated with the queue-sharing group. For an example of setting up this technology, see "Defining an LU6.2 connection for z/OS using APPC/MVS" on page 1557.

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: z/OS

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 1558) that is triggered to start channel `QSG.TO.QM2`. To create a shared transmission queue, use the IBM MQ commands (MQSC) as shown in the following example:

```
DEFINE QLOCAL(QM2) DESCR('Transmission queue to QM2') +
USAGE(XMITQ) QSGDISP(SHARED) +
CFSTRUCT(APPLICATION1) INITQ(SYSTEM.CHANNEL.INITQ) +
TRIGGER TRIGDATA(QSG.TO.QM2)
```

Message channel agents for queue-sharing groups:

A channel can only be started on a channel initiator if it has access to a channel definition for a channel with that name.

A message channel agent is an IBM MQ program that controls the sending and receiving of messages. Message channel agents move messages from one queue manager to another; there is one message channel agent at each end of a channel.

A channel definition can be defined to be private to a queue manager or stored on the shared repository and available anywhere (a group definition). This means that a group defined channel is available on any channel initiator in the queue-sharing group.

Note: The private copy of the group definition can be changed or deleted.

To create group channel definitions, use the IBM MQ commands (MQSC) as shown in the following examples:

```
DEFINE CHL(QSG.TO.QM2) CHLTYPE(SDR) +
TRPTYPE(TCP) CONNAME(QM2.MACH.IBM.COM) +
XMITQ(QM2) QSGDISP(GROUP)
```

```
DEFINE CHL(QM2.TO.QSG) CHLTYPE(RCVR) TRPTYPE(TCP) +
QSGDISP(GROUP)
```

There are two perspectives from which to look at the message channel agents used for distributed queuing with queue-sharing groups:

Inbound

An inbound channel is a shared channel if it is connected to the queue manager through the group listener. It is connected either through the generic interface to the queue-sharing group, then directed to a queue manager within the group, or targeted at the group port of a specific queue manager or the luname used by the group listener.

Outbound

An outbound channel is a shared channel if it moves messages from a shared transmission queue. In the example commands, sender channel `QSG.TO.QM2` is a shared channel because its transmission queue, QM2 is defined with QSGDISP(SHARED).

Synchronization queue for queue-sharing groups:

Shared channels have their own shared synchronization queue called SYSTEM.QSG.CHANNEL.SYNCQ.

This synchronization queue is accessible to any member of the queue-sharing group. (Private channels continue to use the private synchronization queue. See “Defining IBM MQ objects” on page 1533). 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 1558.

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 1063.

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 1554.

Sending end

The connection name (CONNAME) field in the channel definition to connect to your queue-sharing group must be set to the generic interface of your queue-sharing group (see Queue-sharing groups). For more details, refer to Using Sysplex Distributor.

Receiving on TCP using a queue-sharing group

Receiving shared channel programs are started in response to a startup request from the sending channel. To do so, a listener must be started to detect incoming network requests and start the associated channel. You start this listener program with the START LISTENER command, using the inbound disposition of the group, or using the operations and control panels.

All group listeners in the queue-sharing group must be listening on the same port. If you have more than one channel initiator running on a single MVS image you can define virtual IP addresses and start your TCP listener program to only listen on a specific address or host name by specifying IPADDR in the START LISTENER command. (For more information, see START LISTENER.)

Defining an LU 6.2 connection on z/OS:

To define an LU 6.2 connection for a queue-sharing group, certain attributes on the sending and receiving end must be configured.

For information about setting up APPC/MVS, see Setting up communication for z/OS .

Connecting to APPC/MVS (LU 6.2)

The connection name (CONNAME) field in the channel definition to connect to your queue-sharing group must be set to the symbolic destination name, as specified in the side information data set for APPC/MVS. The partner LU specified in this symbolic destination must be the generic resource name. For more details, see Defining yourself to the network using generic resources.

Receiving on LU 6.2 using a generic interface

Receiving shared MCAs are started in response to a startup request from the sending channel. To do so, a group listener program must be started to detect incoming network requests and start the associated channel. The listener program is an APPC/MVS server. You start it with the START LISTENER command, using an inbound disposition group, or using the operations and control panels. You must specify the LU name to use a symbolic destination name defined in the side information data set. For more details, see Defining yourself to the network using generic resources.

Using IBM MQ with IMS

The IBM MQ -IMS adapter, and the IBM MQ - IMS bridge are the two components which allow IBM MQ to interact with IMS.

To configure IBM MQ and IMS to work together, you must complete the following tasks:
- "Setting up the IMS adapter"
- "Setting up the IMS bridge" on page 1572

Related concepts:

"Using IBM MQ with CICS" on page 1573
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 1575
Use this topic if you want to use IMS Open Transaction Manager Access exits with IBM MQ for z/OS.

Related tasks:

"Configuring queue managers on z/OS" on page 1455
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 1573
The actions you must take vary according to whether you use CALLLIBS or LINK, and your version of SMP/E.

Related information:

IBM MQ and IMS

IMS and IMS bridge applications on IBM MQ for z/OS

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 Knowledge Center*.

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 1566.
2. Include the IBM MQ load library thlqual.SCSQAUTH in the JOBLIB or STEPLIB concatenation in the JCL for your IMS control region and for any dependent region that connects to IBM MQ (if it is not in the LPA or link list). If your JOBLIB or STEPLIB is not authorized, also include it in the DFSESL concatenation after the library containing the IMS modules (usually IMS RESLIB).

   Also include thlqual.SCSQANLx (where x is the language letter).

   If DFSESL is present, then SCSQAUTH and SCSQANLx need to be included in the concatenation or added to LNKLIST. Adding to the STEPLIB or JOBLIB concatenation in the JCL is not sufficient.
3. Copy the IBM MQ assembler program CSQQDEFV from thlqual.SCSQASMS to a user library.
4. The supplied program, CSQQDEFV, contains one subsystem name CSQ1 identified as default with an IMS language interface token (LIT) of MQM1. You can retain this name for testing and installation verification.

   For production subsystems, you change the NAME=CSQ1 to your own subsystem name, or use CSQ1. You can add further subsystem definitions as required. See "Defining IBM MQ queue managers to the IMS adapter" on page 1570 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 193 on page 1566.
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 1571.
10. If you are using RACF to protect resources in the OPERCMDS class, ensure that the userid associated with your IBM MQ queue manager address space has authority to issue the MODIFY command to any IMS system to which it might connect.

```
//DYNSTUB EXEC PGM=IEWL,PARM='RENT,REUS,MAP,XREF'
//SYSPRINT DD  SYSOUT=*
//ACSQMOD  DD  DISP=SHR,DSN=thlqual.SCSQLOAD
//IMSLIB   DD  DISP=SHR,DSN=ims.reslib
//SYSLMOD  DD  DISP=SHR,DSN=private.load¹
//SYSUT1   DD  UNIT=SYSDA,SPACE=(CYL,1)
//SYSLIN   DD  *
  INCLUDE ACSQMOD(CSQQSTUB)
  INCLUDE IMSLIB(DFSLI000)
  ALIAS MQCONN,MQCONNX,MQDISC           MQI entry points
  ALIAS MQGET,MQPUT,MQPUT1              MQI entry points
  ALIAS MQOPEN,MQCLOSE                  MQI entry points
  ALIAS MQBACK,MQCMIT                   MQI entry points
  ALIAS CSQBBAK,CSQBCMT                 MQI entry points
  ALIAS MQINQ,MQSET                     MQI entry points
  ALIAS DFSPLI,PLITDLI                  IMS entry points
  ALIAS DFSCOBOL,CBLTDLI                IMS entry points
  ALIAS DFSFOR,FORTDLI                  IMS entry points
  ALIAS DFSASM,ASMTDLI                  IMS entry points
  ALIAS DFSPASCL,PASTDLI                IMS entry points
  ALIAS DFHEI01,DFHEI1                  IMS entry points
  ALIAS DFSAIBLI,AIBTDLI                IMS entry points
  ALIAS DFSESS,DSNWLI,DSNHLI            IMS entry points
  ALIAS MQCRTMH,MQDLTMH,MQDLTMP         IMS entry points
  ALIAS MQINQMP,MQSETMP,MQMHBUF,MQBUFMH   IMS entry points
  MODE AMODE(31),RMODE(24)              Note RMODE setting
  NAME CSQQDYNS(R)
/*

¹Specify the name of a library accessible to IMS applications that
want to make dynamic calls to IBM MQ.
```

Figure 193. Sample JCL to link-edit the dynamic call stub. This includes the IMS language interface module and the IBM MQ IMS stub CSQQSTUB.

Related concepts:

"Setting up the IMS bridge" on page 1572
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.

Related information:

IBM MQ and IMS

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 member, 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 member 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.

An example SSM entry is:

```
CSQ1,MQM1,CSQQESMT,,R,
```

where:

CSQ1	The default subsystem name as supplied with IBM MQ. You can change this to suit your installation.
MQM1	The default LIT as supplied in CSQQDEFV.
CSQQESMT	The external subsystem module name. You must use this value.
R	REO option.

Keyword parameters

IBM MQ parameters can be specified in keyword format; to do this you must specify `SST=Db2`. Other parameters are as described in Positional parameters, and shown in the following example:

```
SST=DB2,SSN=SYS3,LIT=MQM3,ESMT=CSQQESMT
```

where:

SYS3	The subsystem name
MQM3	The LIT as supplied in CSQQDEFV
CSQQESMT	The external subsystem module name

Specifying the SSM EXEC parameter

Specify the SSM EXEC parameter in the startup procedure of the IMS control region. This parameter specifies the one-character to four-character subsystem member name (SSM).

If you specify the SSM for the IMS control region, any dependent region running under the control region can attach to the IBM MQ queue manager named in the IMS.PROCLIB member specified by the SSM parameter. The IMS.PROCLIB member name is the IMS ID (IMSID= *xxxx*) concatenated with the one to four characters specified in the SSM EXEC parameter. The IMS ID is the IMSID parameter of the IMSCTRL generation macro.

IMS lets you define as many external subsystem connections as are required. More than one connection can be defined for different IBM MQ queue managers. All IBM MQ connections must be within the same z/OS system. For a dependent region, you can specify a dependent region SSM or use the one specified for the control region. You can specify different region error options (REOs) in the dependent region SSM member and the control region SSM member. Table 151 on page 1569 shows the different possibilities of SSM specifications.

Table 151. 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.
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	CSQBAPPL		

For more information on the use of IBM MQ classes for JMS, see Using IBM MQ classes for JMS in IMS.

Current releases of IMS support preloading IBM MQ modules from PDS-E format libraries in MPP, BMP, IFP, JMP and JBP regions only. Any other type of IMS region does not support preloading from PDS-E libraries. If preloading is required for any other type of region, then the IBM MQ modules that are provided must be copied to a PDS format library.

Defining IBM MQ queue managers to the IMS adapter:

The names of the IBM MQ queue managers and their corresponding language interface tokens (LITs) must be defined in the queue manager definition table.

Use the supplied CSQQDEFX macro to create the CSQQDEFV load module. Figure 194 shows the syntax of this assembler macro.

```
CSQQDEFX TYPE=ENTRY|DEFAULT,NAME=qmgr-name,LIT=token
or
CSQQDEFX TYPE=END
```

Figure 194. CSQQDEFX macro syntax

Parameters

TYPE=ENTRY|DEFAULT
Specify either TYPE=ENTRY or TYPE=DEFAULT as follows:

TYPE=ENTRY
Specifies that a table entry describing an IBM MQ queue manager available to an IMS application is to be generated. If this is the first entry, the table header is also generated, including a CSQQDEFV CSECT statement.

TYPE=DEFAULT
As for TYPE=ENTRY. The queue manager specified is the default queue manager to be used when MQCONN or MQCONNX specifies a name that is all blanks. There must be only one such entry in the table.

NAME= ***qmgr-name***
Specifies the name of the queue manager, as specified with **MQCONN** or **MQCONNX**.

LIT= token
Specifies the name of the language interface token (LIT) that IMS uses to identify the queue manager.

An MQCONN or MQCONNX call associates the *name* input parameter and the *hconn* output parameter with the name label and, therefore, the LIT in the CSQQDEFV entry. Further IBM MQ calls passing the *hconn* parameter use the LIT from the CSQQDEFV entry identified in the MQCONN or MQCONNX call to direct calls to the IBM MQ queue manager defined in the IMS SSM PROCLIB member with that same LIT.

In summary, the **name** parameter on the MQCONN or MQCONNX call identifies a LIT in CSQQDEFV and the same LIT in the SSM member identifies an IBM MQ queue manager. (For information about the MQCONN call, see MQCONN - Connect queue manager. For information about the MQCONNX call, see MQCONNX - Connect queue manager (extended).)

TYPE=END
Specifies that the table is complete. If this parameter is omitted, TYPE=ENTRY is assumed.

Using the CSQQDEFX macro

Figure 195 on page 1571 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 195. 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 196. 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 197. 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.

Related concepts:

"Setting up the IMS adapter" on page 1564
To use IBM MQ within IMS requires the IBM MQ - IMS adapter (generally referred to as the IMS adapter).

Related information:

IBM MQ and IMS

IMS and IMS bridge applications on IBM MQ for z/OS

Using IBM MQ with CICS

To use IBM MQ with CICS, you must configure the IBM MQ CICS adapter and, optionally, the IBM MQ CICS bridge components.

For more information about configuring the IBM MQ CICS adapter and the IBM MQ CICS bridge components, see the Configuring connections to MQ section of the CICS documentation.

Related concepts:

"Using IBM MQ with IMS" on page 1564
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"
The actions you must take vary according to whether you use CALLLIBS or LINK, and your version of SMP/E.

Related information:

IBM MQ and CICS

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 152. 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 SMPCNTL card specify LINK LMODS CALLLIBS. You can also specify other parameters such as CHECK, RETRY(YES) and RC. See *SMP/E for z/OS: Commands* for further information. 3. Run the SMP/E job.	No action required provided that the SMP/E zones were set up for automatic relinking, and the CSQ8SLDQ job has been run.

Table 152. Service has been applied or the product has been upgraded to a new release (continued)

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
Callable Services	1. Set the Boundary on your SMP/E job to the Target zone. 2. On the SMPCNTL card specify LINK LMODS CALLLIBS. You can also specify other parameters such as CHECK, RETRY(YES) and RC. See *SMP/E for z/OS: Commands* for further information. 3. Run the SMP/E job.	No action required provided that the SMP/E zones were set up for automatic relinking, and the CSQ8SLDQ job has been run.

Table 153. 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 SMPCNTL card specify LINK LMODS CALLLIBS. You can also specify other parameters such as CHECK, RETRY(YES) and RC. See *SMP/E for z/OS: Commands* for further information. 4. Run the SMP/E job.	1. Delete the XZMOD subentries for the following LMOD entries in the IBM MQ for z/OS target zone: CMQXDCST, CMQXRCTL, CMQXSUPR, CSQCBE00, CSQCBE30, CSQCBP00, CSQCBP10, CSQCBR00, CSQUCVX, CSQUDLQH, CSQVXPCB, CSQVXSPT, CSQXDCST, CSQXRCTL, CSQXSUPR, CSQXTCMI, CSQXTCP, CSQXTNSV, CSQ7DRPS, IMQB23IC, IMQB23IM, IMQB23IR, IMQS23IC, IMQS23IM, IMQS23IR 2. Set up the appropriate ZONEINDEXs between the IBM MQ zones and the Language Environment zones. 3. Tailor CSQ8SLDQ to refer to the new zone on the FROMZONE parameter of the LINK commands. CSQ8SLDQ can be found in the SCSQINST library. 4. Run CSQ8SLDQ.
Callable services	1. Change the DDDEF for CSSLIB to point to the new library 2. Set the Boundary on your SMP/E job to the Target zone. 3. On the SMPCNTL card specify LINK LMODS CALLLIBS. You can also specify other parameters such as CHECK, RETRY(YES) and RC. See *SMP/E for z/OS: Commands* for further information. 4. Run the SMP/E job.	1. Delete the XZMOD subentries for the following LMOD entries in the IBM MQ for z/OS target zone: CMQXRCTL, CMQXSUPR, CSQBSRV, CSQILPLM, CSQXJST, CSQXRCTL, CSQXSUPR, CSQ3AMGP, CSQ3EPX, CSQ3REPL 2. Set up the appropriate ZONEINDEXs between the IBM MQ zones and the Callable Services zones. 3. Tailor CSQ8SLDQ to refer to the new zone on the FROMZONE parameter of the LINK commands. CSQ8SLDQ can be found in the SCSQINST library. 4. Run CSQ8SLDQ.

For an example of a job to relink modules when using CALLLIBS, see “Running a LINK CALLLIBS job” on page 1575.

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 198. Example SMP/E LINK CALLLIBS job

Using OTMA exits in IMS

Use this topic if you want to use IMS Open Transaction Manager Access exits with IBM MQ for z/OS.

If you want to send output from an IMS transaction to IBM MQ, and that transaction did not originate in IBM MQ, you need to code one or more IMS OTMA exits.

Similarly if you want to send output to a non-OTMA destination, and the transaction did originate in IBM MQ, you also need to code one or more IMS OTMA exits.

The following exits are available in IMS to enable you to customize processing between IMS and IBM MQ:

- An OTMA pre-routing exit
- A destination resolution user (DRU) exit

OTMA exit names

You must name the pre-routing exit DFSYPRX0. You can name the DRU exit anything, as long as it does not conflict with a module name already in IMS.

Specifying the destination resolution user exit name

You can use the *Druexit* parameter of the OTMACON keyword of the CSQ6SYSP macro to specify the name of the OTMA DRU exit to be run by IMS.

To simplify object identification, consider adopting a naming convention of DRU0xxxx, where xxxx is the name of your IBM MQ queue manager.

If you do not specify the name of a DRU exit in the OTMACON parameter, the default is DFSYDRU0. A sample of this module is supplied by IMS. See the *IMS/ESA® Customization Guide* for information about this.

Naming convention for IMS destination

You need a naming convention for the destination to which you send the output from your IMS program. This is the destination that is set in the CHNG call of your IMS application, or that is preset in the IMS PSB.

A sample scenario for an OTMA exit

Use the following topics for an example of a pre-routing exit and a destination routing exit for IMS:
- “The pre-routing exit DFSYPRX0”
- “The destination resolution user exit” on page 1577

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:

“Using IBM MQ with IMS” on page 1564
The IBM MQ -IMS adapter, and the IBM MQ - IMS bridge are the two components which allow IBM MQ to interact with IMS.

Related information:

IBM MQ and IMS

IMS and IMS bridge applications on IBM MQ for z/OS

The pre-routing exit DFSYPRX0

This topic contains a sample pre-routing exit for OTMA in IMS.

You must first code a pre-routing exit DFSYPRX0. Parameters passed to this routine by IMS are documented in *IMS/ESA Customization Guide*.

This exit tests whether the message is intended for a known OTMA destination (in our example VCPEVCPE). If it is, the exit must check whether the transaction sending the message originated in OTMA. If the message originated in OTMA, it will have an OTMA header, so you should exit from DFSYPRX0 with register 15 set to zero.
- If the transaction sending the message did not originate in OTMA, you must set the client name to be a valid OTMA client. This is the XCF member-name of the IBM MQ queue manager to which you want to send the message. The *IMS/ESA Customization Guide* tells you where to set this value. We suggest you set your client name (in the OTMACON parameter of the CSQ6SYSP macro) is set to the queue manager name. This is the default. You should then exit from DFSYPRX0 setting register 15 to 4.
- If the transaction sending the message originated in OTMA, and the destination is non-OTMA, you should set register 15 to 8 and exit.
- In all other cases, you should set register 15 to zero.

If you set the OTMA client name to one that is not known to IMS, your application CHNG or ISRT call returns an A1 status code.

For an IMS system communicating with more than one IBM MQ queue manager, you should repeat the logic for each IBM MQ queue manager.

Sample assembler code is shown in Figure 199 on page 1577:

```
TITLE 'DFSYPRX0: OTMA PRE-ROUTING USER EXIT'
DFSYPRX0 CSECT
DFSYPRX0 AMODE 31
DFSYPRX0 RMODE ANY
*
SAVE (14,12),,DFSYPRX0&SYSDATE&SYSTIME
SPACE 2
LR  R12,R15        MODULE ADDRESSABILITY
USING DFSYPRX0,R12
*
L   R2,12(,R1)      R2 -> OTMA PREROUTE PARMS
*
LA  R3,48(,R2)      R3 AT ORIGINAL OTMA CLIENT (IF ANY)
CLC  0(16,R3),=XL16'00'  OTMA ORIG?
BNE  OTMAIN        YES, GO TO THAT CODE
*
NOOTMAIN DS 0H            NOT OTMA INPUT
LA  R5,8(,R2)       R5 IS AT THE DESTINATION NAME
CLC  0(8,R5),=C'VCPEVCPE' IS IT THE OTMA UNSOLICITED DEST?
BNE  EXIT0         NO, NORMAL PROCESSING
*
L   R4,80(,R2)      R4 AT ADDR OF OTMA CLIENT
MVC  0(16,R4),=CL16'VCPE' CLIENT OVERRIDE
B   EXIT4         AND EXIT
*
OTMAIN  DS 0H            OTMA INPUT
LA  R5,8(,R2)       R5 IS AT THE DESTINATION NAME
CLC  0(8,R5),=C'VCPEVCPE' IS IT THE OTMA UNSOLICITED DEST?
BNE  EXIT8         NO, NORMAL PROCESSING

*
EXIT0  DS 0H
LA  R15,0         RC = 0
B   BYEBYE
*
EXIT4  DS 0H
LA  R15,4         RC = 4
B   BYEBYE
*
EXIT8  DS 0H
LA  R15,8         RC = 8
B   BYEBYE
*
BYEBYE  DS 0H
RETURN (14,12),,RC=(15)   RETURN WITH RETURN CODE IN R15
SPACE 2
REQUATE
SPACE 2
END
```

Figure 199. OTMA pre-routing exit assembler sample

The destination resolution user exit

This topic contains a sample destination resolution user exit for IMS.

If you have set registers 15 to 4 in DFSYPRX0, or if the source of the transaction was OTMA ***and*** you set Register 15 to zero, your DRU exit is invoked. In this example, the DRU exit name is DRU0VCPE.

The DRU exit checks if the destination is VCPEVCPE. If it is, it sets the OTMA user data (in the OTMA prefix) as follows:

Offset OTMA user data

(decimal)

0	OTMA user data length (in this example, 334)
2	MQMD
326	Reply to format

These offsets are where the IBM MQ - IMS bridge expects to find this information.

We suggest that the DRU exit is as simple as possible. Therefore, in this sample, all messages originating in IMS for a particular IBM MQ queue manager are put to the same IBM MQ queue.

If the message needs to be persistent, IMS must use a synchronized transaction pipe. To do this, the DRU exit must set the OUTPUT flag. For further details, refer to the *IMS/ESA Customization Guide*.

Write an IBM MQ application to process this queue, and use information from the MQMD structure, the MQIIH structure (if present), or the user data, to route each message to its destination.

A sample assembler DRU exit is shown in Figure 200 on page 1579.

```
TITLE 'DRU0VCPE: OTMA DESTINATION RESOLUTION USER EXIT'
DRU0VCPE CSECT
DRU0VCPE AMODE 31
DRU0VCPE RMODE ANY
*
SAVE (14,12),,DRU0VCPE&SYSDATE&SYSTIME
SPACE 2
LR  R12,R15            MODULE ADDRESSABILITY
USING DRU0VCPE,R12
*
L   R2,12(,R1)           R2 -> OTMA DRU PARMS
*
L   R5,88(,R2)           R5 ADDR OF OTMA USERDATA
LA  R6,2(,R5)           R6 ADDR OF MQMD
USING MQMD,R6            AS A BASE
*
LA  R4,MQMD_LENGTH+10       SET THE OTMA USERDATA LEN
STH  R4,0(,R5)           = LL + MQMD + 8
*                       CLEAR REST OF USERDATA
MVI  0(R6),X'00'          ...NULL FIRST BYTE
MVC  1(255,R6),0(R6)        ...AND PROPAGATE IT
MVC  256(MQMD_LENGTH-256+8,R6),255(R6) ...AND PROPAGATE IT
*
VCPE   DS  0H
CLC  44(16,R2),=CL16'VCPE'     IS DESTINATION VCPE?
BNE  EXIT4             NO, THEN DEST IS NON-OTMA
MVC  MQMD_REPLYTOQ,=CL48'IMS.BRIDGE.UNSOLICITED.QUEUE'
MVC  MQMD_REPLYTOQMGR,=CL48'VCPE'  SET QNAME AND QMGRNAME
MVC  MQMD_FORMAT,MQFMT_IMS     SET MQMD FORMAT NAME
MVC  MQMD_LENGTH(8,R6),MQFMT_IMS_VAR_STRING
*                       SET REPLYTO FORMAT NAME
B   EXIT0
*
EXIT0  DS  0H
LA  R15,0             SET RC TO OTMA PROCESS
B   BYEBYE             AND EXIT
*
EXIT4  DS  0H
LA  R15,4             SET RC TO NON-OTMA
B   BYEBYE             AND EXIT
*
BYEBYE  DS  0H
RETURN (14,12),,RC=(15)       RETURN CODE IN R15
SPACE 2
REQUATE
SPACE 2
CMQA  EQUONLY=NO
CMQMDA DSECT=YES
SPACE 2
END
```

Figure 200. Sample assembler DRU exit

Using IBM z/OSMF to automate IBM MQ

V 9.0.1

The IBM z/OS Management Facility (z/OSMF) provides system management functions in a task-oriented, web browser-based user interface with integrated user assistance, so that you can more easily manage the day-to-day operations and administration of your mainframe z/OS systems.

By streamlining some traditional tasks and automating others, z/OSMF can help to simplify some areas of z/OS system management.

Resources can be provisioned or de-provisioned, at a click of a button, from a user provided portal. z/OSMF provides REST APIs to help with this task.

The sample marketplace portal supplied with z/OSMF can also be used to provision and de-provision resources. Alternatively, more experienced users can use the z/OSMF Web User Interface (WUI).

This section assumes that you understand z/OSMF, but if you are unfamiliar with z/OSMF you should read Getting started with z/OSMF. Alternatively, you can access this section from the z/OSMF WUI online help.

You should familiarize yourself with z/OS Cloud configuration, that is:
- Cloud Provisioning - Resource Management and Software Services
- Configuration- Configuration Assistant, and Performance - Workload Management, and
- Performance - Workload Management

Details of these, together with a *Getting Started Tutorial - Cloud*, are in the *What's New in this Release* section.

z/OSMF Version 2.2 introduces role based activities and tasks, so it is important that you understand concepts like:

domains
administrators
approvers
tenants
templates
instances
workflows

and so on. Refer to *Cloud Provisioning* in the *z/OSMF Programming Guide*, or in the z/OSMF WUI help.

Sample IBM MQ z/OSMF workflows and associated files are provided, and can be installed as part of the IBM MQ for z/OS UNIX System Services Components feature. The installation process for this feature, and the directory and file structure, are described in the IBM MQ for z/OS Program Directory, available to download from the IBM Publications Center.

The sample workflows are written in XML and demonstrate how to automate the provisioning (creation) or de-provisioning (destruction) of IBM MQ queue managers, channel initiators, and local queues, and how to perform actions against the provisioned IBM MQ resources. Steps within the workflows submit jobs (JCL), run REXX execs, process Shell scripts, or issue REST API calls.

The samples are designed to illustrate the types of function that can be achieved using z/OSMF. It is anticipated that z/OSMF workflows will generally be used to provision resources and actions like put or get message will, in essence, be performed using IBM MQ applications.

You can run the sample workflows as supplied, provided the workflow variable properties have been set (as discussed in the following sections), or you can customize them as required. You might prefer to write your own workflows to perform additional function. Before running the sample workflows see:
- "Prerequisites" on page 1581
- "Security settings" on page 1582
- "Limitations" on page 1584

Sample workflow applications are provided to:

- “Automate the provisioning or de-provisioning of IBM MQ queue managers and perform actions against the provisioned queue managers” on page 1586
- “Automate the provisioning or de-provisioning of IBM MQ local queues and perform actions against the provisioned queues” on page 1586.

Related concepts:

“Customizing IBM MQ for z/OS” on page 1459
Use this topic as a step by step guide for customizing your IBM MQ system.

Prerequisites

V 9.0.1

The prerequisites you require to run IBM z/OS Management Facility (z/OSMF) with IBM MQ

The IBM MQ workflows shipped in Version 9.0.1 exploit new function in z/OSMF, which is provided through APARs on both z/OS Version 2.1 and Version 2.2. More details are provided in the following text.

1. You have installed and configured IBM z/OS Management Facility Version 2.2 correctly. If you are running with security enabled, ensure that all security settings as documented by z/OSMF have been configured.
2. You have installed the following APARs for:

   z/OS Version 2.1
   - PI71068
   - PI71079
   - PI71082
   - PI71084
   - OA50130

   z/OS Version 2.2
   - PI70526
   - PI70521
   - PI70527
   - PI67839
   - PI70767
   - PI46315
   - OA49081
   - OA49802
   - OA50130
3. The z/OSMF angel (if required) and server processes have been configured.
4. The z/OS Cloud environment has been configured (as briefly discussed above and documented by z/OSMF)
5. IBM MQ for z/OS Version 9.0.1 has been installed and the product load libraries are available.
6. The following IBM MQ queue manager customization tasks have been performed:

Task	Description
1	Identify the z/OS system parameters
2	APF authorize the IBM MQ load libraries
3	Update the z/OS link list and LPA
4	Update the z/OS program properties table

7. The sample workflows and associated files are installed in a suitable UNIX System Services for z/OS (USS) directory.
8. The **'/tmp'** USS directory is available, as the provision.xml workflow might create a temporary file in this directory. If a file is created, the workflow, in general, deletes the file after use.
9. The `deprovision.xml` file has steps in it that invoke the CSQ4ZWS1.rexx and CSQ4ZWS2.rexx REXX execs. These execs wait for the queue manager and channel initiator subsystems to stop; the execs invoke the USS 'SLEEP' command as a system call.

   Depending on your USS configuration, you might find that the 'SLEEP' command does not work as coded. If, during processing you encounter an error which indicates that the 'SLEEP' command cannot be found, you can try replacing the following lines in execs CSQ4ZWS1.rexx and CSQ4ZWS2.rexx:

```
CALL SYSCALLS('ON')       /* Enable USS calls    */
ADDRESS SYSCALL
"SLEEP" 10                           /* Sleep for 10 seconds */
CALL SYSCALLS 'OFF'       /* Disable USS calls    */
```

   with

```
'sleep' 10
```

   Then, issue the Open MVS (OMVS) **`env`** command to check your PATH environment variable setting. Ensure that the directory which contains the **`sleep`** command is defined to the PATH. Note that the **`sleep`** command is typically found in the `/bin` directory.
10. Ensure that z/OSMF has been started.

    Both the angel and server z/OSMF processes must be started and the z/OSMF Web User Interface (WUI) be up and running. For further details, see Liberty profile: Process types on z/OS.

    Even if you intend to drive the workflows using the REST API, the z/OSMF WUI needs to be started. The z/OSMF WUI can be useful for monitoring the creation and execution of workflows.

Related concepts:

"Using IBM z/OSMF to automate IBM MQ" on page 1579
The IBM z/OS Management Facility (z/OSMF) provides system management functions in a task-oriented, web browser-based user interface with integrated user assistance, so that you can more easily manage the day-to-day operations and administration of your mainframe z/OS systems.

Security settings

V 9.0.1

The security settings required to run z/OSMF.

The following User ID variable properties are defined in the properties file. For more details, see "Running the workflows" on page 1589.

User ID property	Description
CSQ_USERID	User ID used to run the workflow steps. Note, however, that selected steps (which generally require an elevated level of authority) will be run with different user IDs based on the setting of the **`CSQ_ADMIN_*`** user IDs listed in the following text. The user ID in use is identified by the **`runAsUser`** property on the respective step in the workflows.
CSQ_ADMIN_APF_USERID	User ID to use when APF authorizing the load library that contains the queue manager system parameter module.
CSQ_APF_APPROVAL_ID	The approval ID used to permit users to run the data set APF authorization step as user CSQ_ADMIN_APF_USERID.
CSQ_ADMIN_CONSOLE_USERID	User ID used when running steps under the run that issue z/OS console commands. **Attention:** This user ID needs to be permitted UPDATE access to the started task profile (MVS.START.STC.*) in the 'OPERCMDS' class. For more details refer to the *z/OS Operator Console Operations* section in the IBM z/OS Knowledge Center.

User ID property	Description
CSQ_CONSOLE_APPROVAL_ID	The approval ID used to permit users to run steps that issue z/OS console commands under the run as user CSQ_ADMIN_CONSOLE_USERID.
CSQ_ADMIN_SAF_USERID	User ID to use when issuing SAF commands.
CSQ_SAF_APPROVAL_ID	The approval ID used to permit users to run the SAF command steps under the run as user CSQ_ADMIN_SAF_USERID.
CSQ_ADMIN_SSI_USERID	User ID to use when issuing the SETSSI command to identify the subsystem being provisioned to z/OS.
CSQ_SSI_APPROVAL_ID	The approval ID used to permit users to run the SETSSI command step under the run as user CSQ_ADMIN_SSI_USERID.

Note: The User ID being used to run the provision and de-provision workflows needs to have sufficient authority as listed below:

1. The Queue Manager provision and de-provision workflows use the SETPROG command to APF authorize data sets. Either the user ID is set in property CSQ_ADMIN_APF_USERID, or the user ID being used to run the workflows needs to be permitted to issue this command. You can achieve this by issuing the following command:

```
PERMIT MVS.SETPROG CLASS(OPERCMDS) ID(value of CSQ_ADMIN_APF_USERID) ACCESS(UPDATE)
```

   Note: The SETPROG command might not persist across an IPL of a z/OS system so, it might be necessary to manually issue the following SETPROG command following an IPL:

```
SETPROG APF,ADD,DSN=value of CSQ_AUTH_LIB_HLQ.value of CSQ_SSID.APF.LOAD,SMS
```

   For more details about the SETPROG command, see Using RACF to control APF lists.

   In addition, you might have enabled FACILITY class to control which libraries can be APF authorized, so you might need to issue the command:

```
PERMIT CSVAPF.libname CLASS(FACILITY) ID(value of CSQ_ADMIN_APF_USERID)
       ACCESS(UPDATE)
```

2. A step in the Queue Manager provision workflow issues the SETSSI command to identify the IBM MQ subsystem to z/OS. The User ID set in property CSQ_ADMIN_SSI_USERID needs to be permitted to use this command. You can achieve this by issuing the following command:

```
PERMIT MVS.SETSSI.ADD CLASS(OPERCMDS) ID(value of CSQ_ADMIN_SSI_USERID)
        ACCESS(CONTROL)
```

   Note: Subsystems that have been identified to z/OS through the SETSSI command do not persist across an IPL of a z/OS system. So, it might be necessary to manually issue the following SETSSI command following an IPL:

```
SETSSI  ADD,S='value of CSQ_SSID',I=CSQ3INI,
       P='CSQ3EPX,value of CSQ_CMD_PFX,S'
```

   For more details about the SETSSI command, see: MVS commands, RACF access authorities, and resource names.

3. The workflows issue queue manager commands, so if you are planning to enable security, the user ID set in property CSQ_ADMIN_RACF_USERID (or the user ID being used to run the workflows) needs to be granted CLAUTH (client authentication) authority to the MQADMIN or the MXADMIN class (depending on which class is being used). This is to allow this user ID to define security profiles to these classes. You can achieve this by issuing the following command:

```
ALTUSR value of CSQ_ADMIN_RACF_USERID CLAUTH(MQADMIN)
```

   For more details about `CLAUTH` see The CLAUTH (class authority) attribute.

4. The deprovision.xml workflow issues z/OS commands, for example, DISPLAY ACTIVE jobs, CANCEL or FORCE subsystems, so the user ID set in property CSQ_ADMIN_CONSOLE_USERID (or the user ID being used to run the workflows) needs to have suitable authority to issue such commands.

5. Users requesting a queue manager instance, using the templates table of the Software Services task, must have permission to access z/OSMF and the Configuration Assistant, as defined by z/OSMF.
6. The user ID of the consumer provisioning a queue manager requires authority to add and delete members from the PROCLIB data set defined with variable CSQ_PROC_LIB.
7. A queue manager must be provisioned ahead of provisioning queues.
8. To use the `queueLoad.xml` and `queueOffload.xml` workflows, the data sets used need to be defined ahead of time. Also, the user ID used to run these workflows needs to be granted UPDATE authority to the data sets.
9. A step in the queue manager `provision.xml` workflow currently disables subsystem security. You can modify Job `csq4znse.jcl` to enable subsystem security by adding the appropriate security commands for protecting IBM MQ resources. However, note that if you do add additional commands, you also need to add commands to delete security permissions in `csq4dse.jcl`, which is submitted by the deprovision.xml workflow.

   Note: This step issues RACF security commands. If you are using an alternate security product, you need to modify this step to issue the appropriate commands for your security product.

Network Requirements

When adding a queue manager template, and resources for the template, you need to click **Create network resource pool**. This creates a resource pool with network resources for this template.

Using the Configuration Assistant, your network administrator needs to complete this network resource pool definition by defining a limit for the number of ports that are to be allocated for this template.

For each template instance, the `provision.xml` workflow allocates a port in the range, and starts a listener to listen on that port.

Classifying with IBM Workload Manager

If you want to classify the queue manager and channel initiator address spaces with WLM, you need to specify this when adding a template for provisioning a queue manager.

Whether to classify or not, is controlled by flags **`CSQ_DEFINE_MSTR_WLM_RULE`** and **`CSQ_DEFINE_CHIN_WLM_RULE`**, which are set in file `workflow_variables.properties`.

For more information about classifying with WLM, refer to the *z/OSMF Configuration Guide*.

Related concepts:
"Prerequisites" on page 1581
The prerequisites you require to run IBM z/OS Management Facility (z/OSMF) with IBM MQ

Limitations

V 9.0.1

Limitations when using z/OSMF with IBM MQ.

1. The `provision.xml` workflow currently automates the following highlighted queue manager customization tasks:

Task	Description
1	Identify the z/OS system parameters
2	APF authorize the IBM MQ load libraries **(provision.xml does APF authorize some libraries)**
3	Update the z/OS link list and LPA
4	Update the z/OS program properties table

Task	Description
5	**Define the IBM MQ subsystem to z/OS**
6	**Create procedures for the IBM MQ queue manager**
7	**Create procedures for the channel initiator**
8	**Define the IBM MQ subsystem to a z/OS WLM service class**
9	Select and set up your coupling facility offload storage environment
10	Set up the coupling facility
11	Implement your ESM security controls
12	Update SYS1.PARMLIB members
13	**Customize the initialization input data sets**
14	**Create the bootstrap and log data sets**
15	**Define your page sets**
16	Add the IBM MQ entries to the Db2 data-sharing group
17	**Tailor your system parameter modules (some)**
18	**Tailor the channel initiator parameters (some)**
19	Set up Batch, TSO, and RRS adapters
20	Set up the operations and control panels
21	Include the IBM MQ dump formatting member
22	Suppress information messages
23	Update your system DIAG member for Advanced Message Security
24	Create procedures for Advanced Message Security
25	Set up the started task user Advanced Message Security
26	Grant RACDCERT permissions to the security administrator for Advanced Message Security
27	Grant users resource permissions for Advanced Message Security

2. Customization tasks that are not highlighted in bold text need to be performed manually, if required.
3. The sample INP1 and INP2 members are currently used as is. If required, additional properties can be defined to control the resources defined by these members.
4. Comments pertaining to specific properties listed in the properties file indicate any limitations of using those properties. For more details, see “Running the workflows” on page 1589.

Related concepts:

“Security settings” on page 1582
The security settings required to run z/OSMF.

Automate the provisioning of IBM MQ objects

z/OS V 9.0.1

Samples are supplied to automate the provisioning of queue managers and local queues.

Automate the provisioning or de-provisioning of IBM MQ queue managers and perform actions against the provisioned queue managers

The following queue manager specific sample z/OSMF workflows are provided:

Workflow name	Description
provision.xml	Provision an IBM MQ for z/OS queue manager This sample workflow: • Provisions the required system resources for a queue manager. • Provisions the required system resources for a channel initiator. • Starts the queue manager (which also starts the channel initiator and TCP/IP listener) • Runs the sample queue manager installation verification program. An environment property can be set to control the provisioning of queue managers with different characteristics. For more information, see "Running the workflows" on page 1589. **Note:** A manifest file (`provision.mf`) is provided to assist with adding a template for this workflow. This file contains a reference to the **qaas_readme.pdf** file which contains additional information. You can access the file through a link, once the template has been added.
deprovision.xml	De-provision an IBM MQ for z/OS queue manager This sample workflow: • Stops the channel initiator (which also stops the TCP/IP listener) and the queue manager. • Waits for the subsystems to stop • De-provisions all channel initiator and queue manager system resources.
startQMgr.xml	Start an IBM MQ for z/OS queue manager This sample workflow starts the queue manager (which also starts the channel initiator and TCP/IP listener).
stopQMgr.xml	Stop an IBM MQ for z/OS queue manager This sample workflow stops the channel initiator (which also stops the TCP/IP listener) and the queue manager.

Each workflow performs one or more steps. Comments in the workflows explain the function performed by each step. Some of the steps just request data input, while some steps submit JCL, invoke REXX execs, Shell scripts, or issue REST API calls to accomplish the stated function.

Refer to each step for the exact name of the JCL or REXX exec files. The workflows and associated JCL or REXX exec files reference variables that are declared in one or more variable XML files. For more details, see "Workflow variable declaration files" on page 1588.

`deprovision`, **`startQMgr`**, and **`stopQMgr`** can be performed as actions against a provisioned IBM MQ for z/OS queue manager.

Automate the provisioning or de-provisioning of IBM MQ local queues and perform actions against the provisioned queues

The following queue specific sample z/OSMF workflows are provided:

Workflow name	Description
defineQueue.xml	Define a local queue This sample workflow demonstrates how z/OSMF workflows can be used to define small, medium, or large sized queues based on property settings. **Note:** A manifest file (`provision.mf`) is provided to assist with adding a template for this workflow. This file contains a reference to the **qaas_readme.pdf** file which contains additional information. You can access the file through a link, once the template has been added.

Workflow name	Description
displayQueue.xml	Display selected attributes of a local queue This sample workflow displays selected attributes of a local queue. The attributes are returned in a z/OSMF variable (refer to the steps in the workflow for the name of the variable) and subsequently displayed. If required, the contents of the variable can be accessed using a REST API. For more details, refer to the *REST APIs for Cloud Provisioning* documented in the z/OSMF Programming Guide, and also see z/OSMF workflow services.
deleteQueue.xml	Delete a local queue This sample workflow deletes a local queue on a specified queue manager.
putQueue.xml	Put one or more messages to a local queue. This sample workflow puts one or more messages to a local queue. The message text can be specified but if more than one message is put to a local queue at the same time, the same message text is used.
getQueue.xml	Get one or more messages from a local queue. This sample workflow gets one or more messages from a local queue. The messages are returned in a z/OSMF variable (refer to the steps in the workflow for the name of the variable) and subsequently displayed. If required, you can access the contents of the variable using a REST API. For more details, refer to the *REST APIs for Cloud Provisioning* documented in the z/OSMF Programming Guide, and also see z/OSMF workflow services.
loadQueue.xml	Load messages from a data set to a local queue. This sample workflow loads messages from a data set on to a local queue. The default name of the data set is specified by setting a property. For more details, see "Running the workflows" on page 1589.
offloadQueue.xml	Offload messages from a local queue to a data set. This sample workflow off-loads messages from a local queue to a data set. The default name of the data set is specified by setting a property. For more details, see "Running the workflows" on page 1589.
clearQueue.xml	Clear messages on a local queue. This sample workflow clears (deletes) all messages on a local queue.

Notes:

1. The **Put Queue** action allows you to enter some message data and put one or more messages onto a queue. If more than one message is to be placed onto a queue during a given request, the same message data is used.
2. The loadQueue.xml and offloadQueue.xml workflows invoke the IBM MQ for z/OS QLOAD utility which is essentially the **dmpmqmsg** utility available with IBM MQ for Multiplatforms. Therefore messages loaded from a data set onto a queue or from a queue onto a data set are expected to be in the **dmpmqmsg** format.

   The easiest way to try out the loadQueue and offloadQueue actions is to do the following:

   a. Issue **putQueue** a few times to put some messages on to a queue.
   b. Use **offloadQueue** to offload the messages from the queue on to a data set.
   c. If required, issue **clearQueue** to remove all messages from the queue.
   d. Use **loadQueue** to load the messages from a data set onto the same or a different queue.

   If you are interested in the **dmpmqmsg** format, you can browse the contents of the data set, once you have issued an Offload request.

3. You can perform **displayQueue**, **deleteQueue**, **putQueue**, **getQueue**, **loadQueue**, **offloadQueue**, and **clearQueue** as actions against a provisioned IBM MQ for z/OS local queue. For further details about actions and action files, refer to the *z/OSMF Programming Guide*.
4. All action related workflows are deleted by default. The reason for this is to minimize the need for users to cleanup workflows.

   The problem with this however is that where an action results in some output. For example, the **displayQueue** and **getQueue** actions both produce output.

   The output cannot be seen since the related workflow is deleted as soon as the action has been performed. So, if you drive the workflow actions from the z/OS WUI, you need to set the **cleanAfterComplete** flag to *false* on the **<workflow>** tag for each action whose output you want to see.

   For example, to see the output of **displayQueue**, set the flag as follows:

```
<action name="displayQueue">
  <workflow cleanAfterComplete="false">
  ...

  ...
  </workflow>
</action>
```

   However, this means that you then have to manually clean up action related workflows.

Each sample z/OSMF workflow performs one or more steps. Comments in the workflows explain the function performed by each step. Some of the steps just request data input while some steps submit JCL and others invoke REXX execs to accomplish the stated function.

Refer to each step for the exact name of the JCL or REXX exec files. The workflows and associated JCL or REXX exec files reference variables that are declared in one or more “Workflow variable declaration files.”

Related concepts:

“Limitations” on page 1584
Limitations when using z/OSMF with IBM MQ.

Running workflows

V 9.0.1

A description of the files referenced by the sample The z/OSMF workflows, and how you run a workflow.

Workflow variable declaration files

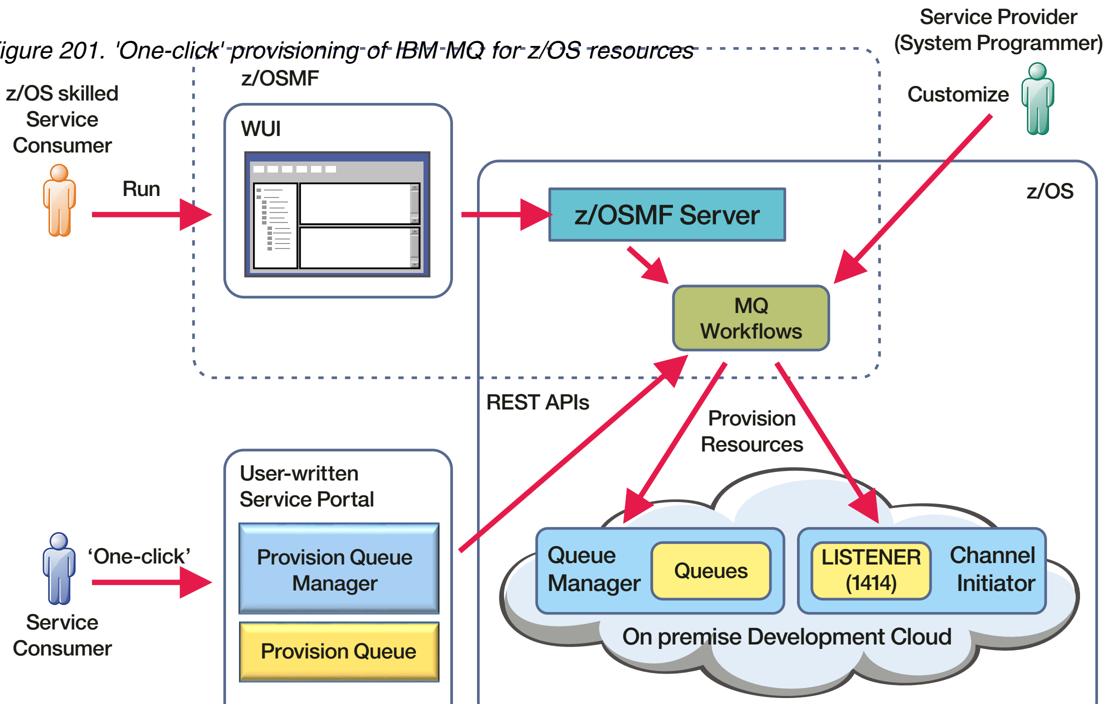
The following files declare variables that are referenced by the sample z/OSMF workflows and associated JCL or REXX exec files:

Workflow variable declaration file name	Description
common_variables.xml	Variables common to both the queue manager (plus channel initiator) and queue workflows.
qmgr_variables.xml	Variables specific to the queue manager (plus channel initiator) workflows.
queue_variables.xml	Variables specific to the queue workflows.
tcpip_variables.xml	Variables specific to the queue manager (plus channel initiator) workflows, and used for identifying TCP/IP resources.

Note: The default visibility of variables is *private*. To allow variables to be queried using the z/OSMF REST API, selected variables have been marked as *public*. However, you can change the visibility of a given variable if required.

Running the workflows

Figure 201. 'One-click' provisioning of IBM MQ for z/OS resources



Before the workflows can be run, some properties need to be set in the following file:

Workflow variable properties file name	Description
workflow_variables.properties	Initial properties for the workflow variables. Comments in the file indicate the purpose of each property. • Properties within meta-brackets (< >) need to be set to user specific values. • An environment property can be set to provision queue managers for development (DEV), or test (TEST), or quality assurance (QA), or production (PROD) environments. Additional property settings control the characteristics of the queue manager to be provisioned for each environment. For example you can vary the number of active logs, or the number of pagesets, for each environment type. • Other properties are set to IBM MQ default values but can be modified to meet local conventions if required.

In general, once the properties have been set, the workflows can be run as is. However, if required, you can customize a workflow to modify or remove existing steps, or to add new steps.

Workflows can be run:

- From the z/OSMF WUI.

  From Cloud Provisioning -> Software Services in the WUI, workflows can be run in automatic or manual mode. The manual mode is useful when testing, and in both modes the progress of each step in the workflow can be monitored.

  For more details, refer to *Cloud Provisioning* in the z/OSMF WUI help, and also see Creating a workflow.
- Using the z/OSMF REST Workflow Services.

  The REST Workflow Services can be used to run workflows through a REST API. This mode is useful for creating one-click operations from a user-written portal.

For more details, refer to details of the *REST APIs for Cloud Provisioning* documented in the z/OSMF Programming Guide, and also see z/OSMF workflow services.

- Using the sample marketplace portal provided with z/OSMF.

Related concepts:

"Automate the provisioning of IBM MQ objects" on page 1585
Samples are supplied to automate the provisioning of queue managers and local queues.

IBM MQ Console on z/OS security definitions

You need to create some security definitions for your external security manager (ESM). The following text assumes that you are using RACF. Some of the definitions are required for WebSphere Application Server Liberty (Liberty), and some are IBM MQ specific.

This information has moved. See Configuring System Authorization Facility interface for more information.

Setting up the IBM MQ Console on z/OS

V 9.0.2

How you set up the files for the IBM MQ Console.

Before you begin

You should have read Planning to use the IBM MQ Console on z/OS and carried out the procedures in Configuring System Authorization Facility interface

About this task

You need to put the IBM MQ files on the z/OS machine, together with the SCSQAUTH, SCSQLOAD, and SCSQANL* libraries needed to run IBM MQ.

You will also need the USS files.

Procedure

1. If the installed files are not available on the LPAR where IBM MQ will be running, you need to copy the libraries and files. There are different ways to copy the files:
   a. Optional: You can copy the data set containing the USS file system containing the IBM MQ files, and mount this on the z/OS image.
   b. Optional: FTP the files `CSQ8WPX1.pax` and `CSQ8JTR1.tar` in binary and unpack them:
      1) Create the directory for your IBM MQ files, for example by using the command `mkdir /mqm/v901`.
      2) You can use the `df -P /mqm/v901` command to display the free and available space in the file system containing `/mqm/v901`.
      3) Change into this directory by issuing the command `cd /mqm/v901`.
      4) Expand the `pax` file, for example, by issuing the following command:
         ```
         pax -ppx -sm^wlpmwebm -rf location_of_the_CSQ8WPX1.pax
         ```
      5) Expand the tar file, for example, by issuing the following command:
         ```
         tar -xpoXf location_of_the_CSQ8JTR1.tar
         ```
      6) Remove files CSQ8WPX1.pax and CSQ8JTR1.tar.
2. On each queue manager that REST API is to be issued to, define a reply to queue for use by the IBM MQ REST API implementation. See the definition of QLOCAL('SYSTEM.REST.REPLY.QUEUE') in CSQ4INSG.JCL.

What to do next

Configure the IBM MQ Console without security

Configuring the IBM MQ Console on z/OS without security

V 9.0.2

An initial setup of the IBM MQ Console, using HTTP, and without TLS.

Before you begin

You have completed the procedures in "Setting up the IBM MQ Console on z/OS" on page 1590.

About this task

A Liberty server is configured using XML files in USS. The mqweb server is shipped with a `server.xml` and `amqwebuser.xml` file. The `mqwebuser.xml` file is where you place your customized configuration for the mqweb server.

Note: You should not edit the `server.xml` file.

This section describes how to set up the IBM MQ Console using HTTP only, without any security checking. The section on setting up security describes how to add security and TLS.

Procedure

1. Follow the instructions to create the IBM WLP server definition.
2. Change into the directory for your configuration. For example, issue the following command:
   ```
   cd /u/mqm/mqconsole/servers/mqweb
   ```
3. Rename the existing `mqwebuser` file by issuing the following command:
   ```
   mv  mqwebuser.xml mqwebuser.xml.old
   ```
   There are several example files you can use as a basis for your configuration. You might find it easier to use the simplest configuration, and add function to that configuration.
4. Copy one of the example files from the IBM MQ installation directory to your Liberty user directory, using the following command:
   ```
   cp  PathPrefix/web/mq/samp/configuration/no_security.xml   mqwebuser.xml
   ```
   where `PathPrefix` is the IBM MQ Unix System Services Components installation path. Note that the `no_security.xml` file provides HTTP with no authentication or role mapping.
5. Make this file readable for other user IDs in the group, by issuing the following command:
   ```
   chmod 750 mqwebuser.xml
   ```
6. Edit this file, for example, by using **`oedit`**. The default port is 9080. If you need to change it, change the value in the `<variable name="httpPort" value="9080"/>` statement.

   Note that the variables **`httpHost`**, **`httpPort`**, and **`httpsPort`**, are used in the Liberty **`httpEndpoint`** statement.

   See httpEndpoint for further information.

   The **`httpHost`** variable is used in the `httpEndPoint host=` statement.

   You can replace **`localhost`** in `<variable name="httpHost" value="localhost"/>` with a specific TCP/IP host name. If you are using multiple TCP stacks, specify the host name of the stack you want to use.

   If you want to use `https`, insert `<variable name="httpsPort" value="9443"/>` and specify the port number for TLS.
7. Follow the instructions to create a procedure for the IBM WLP server .

There is a reference to java JAVA_HOME, for example, `JAVA_HOME=/java/J64/`

Take your JAVA_HOME variable, put `/bin/classic` on the end, and use the **`ls`** command to check the directory exists, for example: `ls /java/J64/bin/classic` This command lists the files in the directory, for example `libjvm.x`.

If the directory or file is not found, check your JAVA_HOME statement.

What to do next

Getting started with the IBM MQ Console

Configuring IBM MQ Advanced for z/OS VUE

z/OS MQ Adv. VUE

Use this information to configure features that are available with IBM MQ Advanced for z/OS VUE entitlement.

About this task

From Version 9.0.3, you can use the features provided in the IBM MQ Advanced for z/OS VUE Connector Pack to simplify MFT topology on z/OS and make use of the connectivity from IBM MQ Advanced for z/OS, Value Unit Edition queue managers to two services in IBM Cloud (formerly Bluemix): IBM Cloud Product Insights and IBM Blockchain.

V 9.0.4 From Version 9.0.4, you can connect a IBM MQ classes for JMS, or IBM MQ classes for Java, application to a queue manager on z/OS, that has the **ADVCAP**(ENABLED) attribute, by using a client connection.

Procedure

1. Configure IBM MQ Advanced for z/OS, Value Unit Edition for use with IBM Cloud Product Insights service in IBM Cloud (formerly Bluemix).
2. Enable Managed File Transfer agent remote connections with IBM MQ Advanced for z/OS, Value Unit Edition.
3. Configure IBM MQ Advanced for z/OS VUE for use with the IBM Blockchain service in IBM Cloud.
4.

Configuring IBM MQ Advanced for z/OS VUE for use with IBM Cloud Product Insights service in IBM Cloud (formerly Bluemix)

z/OS V 9.0.3 MQ Adv. VUE

Connect your IBM MQ Advanced for z/OS, Value Unit Edition queue manager to the IBM Cloud Product Insights service in IBM Cloud (formerly Bluemix) to report and view startup and usage information.

Before you begin

Before you configure your IBM MQ queue managers to use a IBM Cloud service, you must have an IBM Cloud (formerly Bluemix) account. To create your account, see Sign up for IBM Cloud.

About this task

At IBM MQ Version 9.0.3 you can configure and connect your IBM products with the IBM Cloud Product Insights service in IBM Cloud and see all the registered products in your organization in a single dashboard.

When a queue manager is configured to connect to an instance of the Product Insights service, the following information is reported to IBM Cloud (formerly Bluemix).

Registration details:
- IBM MQ for z/OS host name, that is the LPAR name.
- IBM MQ for z/OS queue manager name.
- IBM MQ for z/OS instance identifier; that is the queue manager UUID.
- Persistent ID, unique identifier related to the product name.
- IBM MQ for z/OS installation directory allocated for the program, which is always `/usr/lpp/mqm`.
- Operating system name, z/OS.
- Operating system version.
- Last Started time.
- Products, which is an array of products installed that relate to this queue manager. Each product is identified by the following fields:
  - Product name of IBM MQ Advanced for z/OS, Value Unit Edition
  - Product version, for example, 9.0.3
  - Persistent ID

Usage data

Periodically, usage data is recorded, which includes the following additional fields:
- Start time; the start of the usage window.
- End time; the end of the usage window.
- Usage data, which is usage metrics that has been measured over the start time until the end time period.

  The metrics that IBM MQ Advanced for z/OS, Value Unit Edition uploads are:
  - Count of persistent puts
  - Count of non-persistent puts
  - Count of persistent bytes put
  - Count of non-persistent bytes put

  Note: These metrics are put out to the System Management Facility (SMF), only if SMF Class1 statistics trace is enabled. See Using System Management Facility for more information.

  This allows you to compare the data in SMF with the data in IBM Cloud (formerly Bluemix).

  The SMF statistics are collected, as governed by the queue manager STATIME property. To reduce interactions with the IBM Cloud service, data is sent no more frequently than once every 15 minutes.

You can monitor your queue manager usage metrics in your Product Insights service instance dashboard.

For more information, see IBM Cloud Product Insights service in IBM Cloud (formerly Bluemix) and the IBM Cloud Product Insights developer center.

Results

You have created a Product Insights service instance, and configured your queue manager to connect to the instance, when you have completed the tasks in the next two topics.

You can see the information about your queue manager in the Product Insights service instance dashboard.

Related tasks:

"Creating an IBM Cloud Product Insights service instance on IBM Cloud (formerly Bluemix)"
Use your IBM Cloud account to create a Product Insights service instance and copy the security credentials that you need to register your queue managers to the service.

"Configuring a z/OS queue manager for use with the IBM Cloud Product Insights service instance on IBM Cloud (formerly Bluemix)" on page 1595
You can carry out this process only if your enterprise is using IBM MQ Advanced for z/OS, Value Unit Edition. Your IBM Cloud Product Insights service instance dashboard shows data for only the queue managers that are configured to include the security and IBM Cloud registration information.

Creating an IBM Cloud Product Insights service instance on IBM Cloud (formerly Bluemix)

z/OS V 9.0.3 MQ Adv. VUE

Use your IBM Cloud account to create a Product Insights service instance and copy the security credentials that you need to register your queue managers to the service.

About this task

The Product Insights service instance that you are creating is associated to a single organization and space within IBM Cloud and has unique credentials. You must create at least one organization and space but if you need to separate the data you can create multiple spaces within an organization and have a service in each space.

AIX Linux Windows See "Creating an IBM Cloud Product Insights service instance on IBM Cloud (formerly Bluemix)" on page 1425, for information on carrying out this process on AIX, Linux, and Windows.

Procedure

1. Create an instance of the IBM Cloud Product Insights service by following these steps:
   a. Log in to IBM Cloud.
   b. In the menu bar of the IBM Cloud user interface, click **Catalog**.
   c. Click **Integrate** in the **Platform** section, then click the **Product Insights** tile. The Product Insights service page is displayed.
   d. Optional: Specify a service name. The **Service** name field is available when you are logged in.
   e. Optional: Specify a name for the credentials that you want to use for the connection with IBM MQ. The **Credentials** name field is available when you are logged in.
   f. Click **Create**. Your IBM Cloud Product Insights service instance is created and the **Getting Started** tab of your Product Insights service dashboard is shown.
2. Copy the security credentials for connecting your queue manager to the IBM Cloud Product Insights service by following these steps:
   a. Click the **Service credentials** tab. The list of credentials for your on-premises products is displayed and includes the service credentials that you specified for your new Product Insights service.
   b. Click **View credentials** to show the API host and the API key values that you need to use to connect your IBM MQ queue manager to the Product Insights service instance.

c. Copy the API host from the **Service credentials** tab, so that you can add it to the CSQMQMIN data set for your queue manager. You need this in the next task.

d. Copy the API key from the **Service credentials** tab so that you can add it to the CSQMQMIN data set for your queue manager. You need this in the next task.

What to do next

Follow the instructions for "Configuring a z/OS queue manager for use with the IBM Cloud Product Insights service instance on IBM Cloud (formerly Bluemix)."

Configuring a z/OS queue manager for use with the IBM Cloud Product Insights service instance on IBM Cloud (formerly Bluemix)

z/OS V 9.0.3 MQ Adv.VUE

You can carry out this process only if your enterprise is using IBM MQ Advanced for z/OS, Value Unit Edition. Your IBM Cloud Product Insights service instance dashboard shows data for only the queue managers that are configured to include the security and IBM Cloud registration information.

Before you begin

You created an IBM Cloud Product Insights service instance as described in "Creating an IBM Cloud Product Insights service instance on IBM Cloud (formerly Bluemix)" on page 1594.

AIX Linux Windows For non-z/OS platforms, see "Configuring a queue manager for use with the IBM Cloud Product Insights service instance on IBM Cloud (formerly Bluemix)" on page 1426.

About this task

Set up security and IBM Cloud registration information for your queue manager, and then connect to the Product Insights service instance that you have already created.

When your queue manager connects to the Product Insights service in IBM Cloud, the IBM Cloud public certificates are required by the queue manager to verify the authenticity of the service.

Procedure

1. Download the `DigiCert Global Root CA` and `DigiCert SHA2 Secure Server CA` certificates from the DigiCert website. website, on a workstation, using one of the following methods:
   a. Use CURL to download the certificates. The following commands save the certificates, with the file name after the **-o** option, in the current directory,

```
curl -o DigiCertGlobalRootCA.crt https://www.digicert.com/CACerts/DigiCertGlobalRootCA.crt
```

   and

```
curl -o DigiCertSHA2SecureServerCA.crt https://www.digicert.com/CACerts/DigiCertSHA2SecureServerCA.crt
```

   b. Use your browser to download the certificates from the DigiCert website.
      - Find the row in the table with the certificate name `DigiCert Global Root CA`, right click **Download**, then **Save link as** or **Save target as**, depending on your browser. Select the location where you want to save the certificate.
      - Find the row in the table with the certificate name `DigiCert SHA2 Secure Server CA`, right click **Download**, then **Save link as** or **Save target as**, depending on your browser. Select the location where you want to save the certificate.

2. Add the certificates to your External Security Manager, for example RACF, on the mainframe, if not already present. See Adding certificate authority certificates for information on the methods you can use.

   Make a note of the labels of the certificates. You will add these certificates to the keyring for your queue manager, in step 5.
3. Create your queue manager, for example *QM01*. If you are upgrading an existing queue manager, review the CSQ4MSTR.JCL sample, which can be found in `hlq.SCSQPROC`. In particular, look at the CSQ4INSC queue definition and the CSQMQMIN DD card. For more information, see "Configuring queue managers on z/OS" on page 1455.

   Ensure that you complete "Configuring the `ReportingService` (formerly `BluemixRegistration`) stanza" on page 1520. This task adds the `ReportingService` stanza to the CSQMQMIN data set.
4. Create a RACF keyring for your queue manager, by issuing the following command:

   ```
   RACDCERT ADDRING(ring-name) ID(ring-owner)
   ```

   for example:

   ```
   RACDCERT ADDRING(QM01RING) ID(USER123)
   ```

   Notes:

   a. You might need to create a key ring if your queue manager does not have one defined. Issue the following command to find out:

      ```
      +cpf DISPLAY QMGR SSLKEYR
      ```

      If the SSLKEYR value is blank, you need to create a RACF keyring for your queue manager, by issuing the following command:

      ```
      RACDCERT ADDRING(ring-name) ID(ring-owner)
      ```

      The user ID in the RACDCERT, needs to have that user ID replaced by the ring-owner.
   b. The ring-owner user ID has to be the same as the user ID that the channel initiator runs under.

   Later in step 7, the keyring needs to be identified to your queue manager.
5. Issue the following commands from TSO, or submit them using JCL.

   Attention: The label on `RACDCERT CONNECT` is the one from step 2

   ```
   RACDCERT ID(ring-owner) CONNECT(CERTAUTH LABEL('global-root-ca')
   RING(ring-name) USAGE(CERTAUTH))

   for example

   RACDCERT ID(USER123) CONNECT(CERTAUTH LABEL('global-root-ca')
   RING(QM01RING) USAGE(CERTAUTH))
   ```

   ```
   RACDCERT ID(ring-owner) CONNECT(CERTAUTH
   LABEL('secure-server-ca') RING(ring-name) USAGE(CERTAUTH))

   for example:

   RACDCERT ID(USER123) CONNECT(CERTAUTH
   LABEL('secure-server-ca') RING(QM01RING) USAGE(CERTAUTH))
   ```

   If required, you can issue the following command to display the contents of the keyring:

   ```
   RACDCERT ID(ring-owner) LISTRING(ring-name)
   ```

   for example:

   ```
   RACDCERT ID(USER123) LISTRING(QM01RING)
   ```
6. Start your queue manager. For more information, see Starting and stopping a queue manager.
7. If you had to create a keyring in step 4, you need to identify the keyring to your queue manager by using the ALTER QMGR command. See Specifying the key repository location for a queue manager for more information.
8. Start your channel initiator. For more information, see START CHINIT.

9. During start-up, the channel initiator establishes connectivity with the IBM Cloud Product Insights service on IBM Cloud and registers with the service. Message CSQX193I is put out to the channel initiator job log if a successful connection has been made. Note that this message is issued only once. Messages CSQX194E and CSQX198E are issued when a connection fails.

   You can view the information about your queue manager in your Product Insights service instance.

   When the reporting status is active, the start-up and usage information is reported to the IBM Cloud Product Insights service. The usage information is updated periodically.
10. To stop your queue manager from reporting to the IBM Cloud Product Insights service:
    a. Turn off SMF class 1 statistics,

       This option stops the reporting temporarily, but also turns off SMF Class1 statistics. or
    b. Comment out the CSQMQMIN DD card in your queue manager start-up procedure; then stop and restart your queue manager and channel initiator.

       This option requires that you restart your system.

What to do next

See Troubleshooting IBM Cloud Product Insights for more information, if you are having problems.

Related tasks:

"Configuring IBM MQ Advanced for z/OS VUE for use with IBM Cloud Product Insights service in IBM Cloud (formerly Bluemix)" on page 1592
Connect your IBM MQ Advanced for z/OS, Value Unit Edition queue manager to the IBM Cloud Product Insights service in IBM Cloud (formerly Bluemix) to report and view startup and usage information.

Related information:

"Configuring the `ReportingService` (formerly `BluemixRegistration`) stanza" on page 1520
You can carry out this process only if your enterprise is using IBM MQ Advanced for z/OS, Value Unit Edition. If you want to publish registration and usage data for your queue manager to the IBM Cloud Product Insights service on IBM Cloud (formerly Bluemix), you need to perform this task once, for each queue manager.

Connecting to IBM Cloud Product Insights in IBM Cloud through an HTTP proxy

z/OS V 9.0.4 MQ Adv. VUE

If your queue manager is running on a system that does not have direct access to the internet, you can use an HTTP proxy that your organization provides to connect to your IBM Cloud Product Insights service instance in IBM Cloud. Alternatively, you might use the gateway that is provided by the Product Insights service.

About this task

Use this task to configure your queue manager to connect to the Product Insights service instance in IBM Cloud through an HTTP proxy that is provided by your enterprise.

You do this by adding a **`ServiceProxy`** attribute to the V 9.0.5 `ReportingService` (formerly `BluemixRegistration`) stanza in the CSQMQMIN dataset.

The **`ServiceProxy`** attribute is optional, so you can still connect your queue manager to a Product Insights service instance through the gateway provided by Product Insights. See the Product Insights developer center information on how to use a Product Insights gateway..

Procedure

1. Add a service proxy attribute to the `ReportingService` stanza in the CSQMQMIN dataset. The **`ServiceProxy`** attribute should have the http:// prefix and can be:

- A host name
- An IPv4 address
- An empty string. This disables the proxy support and an attempt is made to connect directly to IBM Cloud.

Optionally, you can specify a port at the end of the **`ServiceProxy`** value with a colon separating the address and port. If you do not specify the port, 1080 is used.

Important: The **`ServiceProxy`** parameter supports a valid http:// URL only.

The channel initiator supports only the HTTP protocol, when using a proxy to talk to an IBM Cloud service. If you specify a protocol that is not supported, IBM MQ rejects the proxy value with error message CSQM574E.

The queue manager starts, but Product Insights registration is not enabled.

2. Restart the queue manager to read the revised configuration file.

   Notes:
   a. Message CSQM578I is issued when the CSQMQMIN DD card is successfully read.
   b. If you have configured a proxy value that is not valid, you must configure a valid value, and restart the queue manager again.
   c. If the address or host name cannot be resolved by the Toolkit, message CSQX194E, or message CSQX198E is logged.

      In this case, registration is reattempted next time round (that is, 15 minutes later) and either succeeds, if the host can now be resolved, or another CSQX198E error message logged if the problem still exists.

MFT agent connectivity to remote z/OS queue managers

z/OS V 9.0.3 MQ Adv. VUE

Managed File Transfer agents on z/OS, that are running under the product identifier (PID) of IBM MQ Advanced for z/OS VUE, can connect to a remote queue manager on z/OS by using a client connection.

For more information, see Enable agent remote connections with IBM MQ Advanced for z/OS, Value Unit Edition only.

Configuring IBM MQ Advanced for z/OS VUE for use with blockchain

z/OS Linux V 9.0.3 MQ Adv. VUE

Set up and run the IBM MQ Bridge to blockchain to securely connect an IBM MQ on z/OS queue manager and IBM Blockchain. Use the bridge to asynchronously connect to, look up and update the state of a resource in your blockchain, by using a messaging application that connects to your IBM MQ Advanced queue manager.

Before you begin

- IBM MQ Bridge to blockchain is available as a part of a Connector Pack on IBM MQ Advanced for z/OS VUE Version 9.0.4. You can connect to z/OS queue managers that are running on the same command level.
- IBM MQ Bridge to blockchain is supported for use with your blockchain network that is based on Hyperledger Fabric v1.0 architecture.
- The IBM MQ Bridge to blockchain must be installed, configured, and run on an x86 Linux environment that has the following installed:

- IBM MQ Version 9.0.3 Redistributable Java client.
- IBM Java runtime environment Version 8.

If you already have IBM MQ Version 9.0.4 Redistributable Java client and IBM Java runtime environment Version 8 installed, you do not need to complete steps 4 on page 1600 and 5 on page 1601.

About this task

Blockchain is a shared, distributed, digital ledger that consists of a chain of blocks that represent agreed upon transactions between peers in a network. Each block in the chain is linked to the previous block, and so on, back to the first transaction.

IBM Blockchain is built on Hyperledger Fabric and you can develop with it locally with Docker or in a container cluster in IBM Cloud (formerly Bluemix). You can also activate and use your IBM Blockchain network in production, to build, and govern a business network with high levels of security, privacy, and performance. For more information, see the IBM Blockchain Platform.

Hyperledger Fabric is an open source, enterprise blockchain framework that is developed collaboratively by members of the Hyperledger Project, including IBM as the initial code contributor. Hyperledger Project, or Hyperledger, is a Linux Foundation open source, global, collaborative initiative to advance cross-industry blockchain technologies. For more information, see IBM Blockchain, Hyperledger Projects, and Hyperledger Fabric.

If you are already using IBM MQ Advanced for z/OS VUE and IBM Blockchain, you can use the IBM MQ Bridge to blockchain to send simple queries, updates and receive replies from your blockchain network. In this way, you can integrate your on-premises IBM software with the cloud blockchain service.

A brief overview of the bridge operating process can be seen in Figure 1. A user application puts a JSON formatted message on the input/request queue on the z/OS queue manager. The bridge connects to the queue manager, gets the message from the input/request queue, checks that the JSON is correctly formatted, then issues the query or an update to the blockchain. The data that is returned by the blockchain is parsed by the bridge and placed on the reply queue, as defined in the original IBM MQ request message. The user application can connect to the queue manager, get the response message from the reply queue, and use the information.

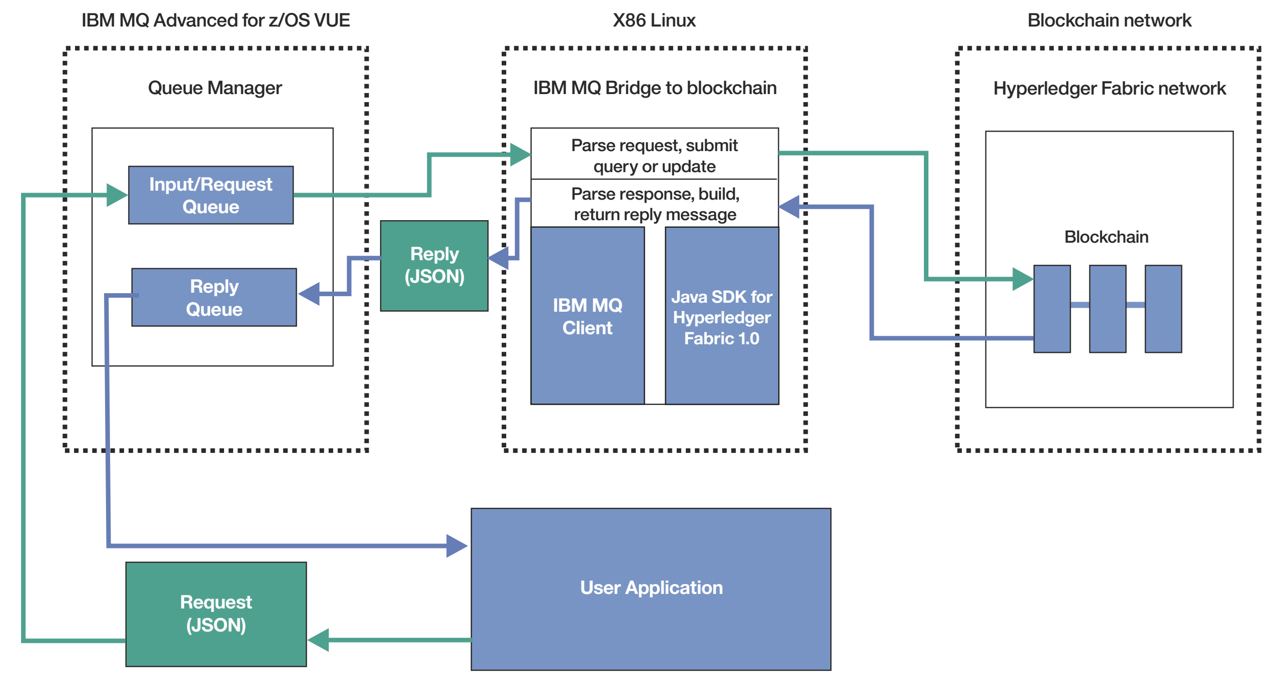

IBM MQ Advanced for z/OS VUE
X86 Linux
Blockchain network
Queue Manager
IBM MQ Bridge to blockchain
Hyperledger Fabric network
Input/Request Queue
Parse request, submit query or update
Parse response, build, return reply message
Reply (JSON)
Reply Queue
IBM MQ Client
Java SDK for Hyperledger Fabric 1.0
Blockchain
User Application
Request (JSON)

Figure 202. IBM MQ Bridge to blockchain

You can configure the IBM MQ Bridge to blockchain to connect to a blockchain network as a participant, or peer. When the bridge is running, a messaging application requests the bridge to drive chaincode routines that query or update the state of the resource and return the results as a response, to the messaging application.

Procedure

1. Create the objects for the bridge that are defined in `csq4bcbq.jcl`. Sample bridge queue definitions are provided for the default named queues that are used for user credentials and for message input to the bridge, `SYSTEM.BLOCKCHAIN.IDENTITY.QUEUE`, and `SYSTEM.BLOCKCHAIN.INPUT.QUEUE`.
   a. Copy `csq4bcbq.jcl` to a z/OS data set.
   b. Edit the `csq4bcbq.jcl` to customize your z/OS queue manager. You must provide a queue manager name and the high level qualifier for the IBM MQ product libraries. You can choose to modify the **`APPL1`** bridge queue examples, or add further `INPUT` and `REPLY` queues for additional applications.
   c. Submit the `csq4bcbq.jcl` to create the objects that you defined.
2. Transfer the `x86download.tar.gz` from the `x86download` directory to your x86 Linux environment by using your preferred method. Ensure that the file is transferred in binary mode.
3. On x86 Linux, unpack the `x86download.tar.gz`
   ```
   tar -xvzf x86download.tar.gz
   ```
   The four directories that are unpacked are `bin`, `lib`, `prereqs`, and `samp`.
4. Download the IBM Java runtime environment Version 8 to your x86 Linux environment.
   a. Click the **Installable package (InstallAnywhere as root)** link on the IBM Java SDK Developer Centre Java 8 Downloads page, with the file name `ibm-java-x86_64-jre-8.0-4.6.bin`, from the **Linux on AMD64/EMT64T** section. The IBM SDK, Java Technology Edition, Version 8 license page is displayed.
   b. Agree to the license to continue. On the download window, select **Save file** for the download to begin.

c. Run the `ibm-java-x86_64-jre-8.0-4.6.bin` file to install it on your x86 Linux environment. The default installation location is the `/opt/ibm/` directory.

```
./ibm-java-x86_64-jre-8.0-4.6.bin
```

d. Set the path to your IBM 8 JRE:

```
export PATH=/opt/ibm/java-x86_64-80/jre/bin:$PATH
```

5. Download the IBM MQ Version 9.0.4 Redistributable Java client from Fix Central.
   a. Click the `9.0.4.0-IBM-MQC-Redist LinuxX64` link.
   b. Select **Download using your browser (HTTPS).** Click continue.
   c. Agree to the terms of the license.
   d. Click the `9.0.4.0-IBM-MQC-Redist-LinuxX64.tar.gz` link and select **Save file** to download it.
   e. Unpack the `9.0.4.0-IBM-MQC-Redist-LinuxX64.tar.gz` to a directory on your x86 Linux environment.
   f. Set the path to the directory where you unpacked the Redistributable Java client.

```
export MQ_JAVA_INSTALL_PATH=/unpack_location/java
```

Results

You transferred the IBM MQ Bridge to blockchain from your z/OS to your x86 Linux environment, installed the IBM JRE 8, and the IBM MQ Version 9.0.4 Redistributable Java client.

What to do next

Use the information for your z/OS queue manager and the credentials from your blockchain network to create a configuration file for the IBM MQ Bridge to blockchain.

Related information:

IBM Blockchain Developer Center

Creating the configuration file for the IBM MQ Bridge to blockchain

V 9.0.3 MQ Adv. VUE

Enter your queue manager and your blockchain network parameters to create the configuration file for the IBM MQ Bridge to blockchain to connect to your IBM MQ and IBM Blockchain networks.

Before you begin

- You created and configured your blockchain network.
- You have the credentials file from your blockchain network.
- You installed the IBM MQ Bridge to blockchain, on your x86 Linux environment.
- You have the IBM MQ Bridge to blockchain, IBM MQ Version 9.0.4 Redistributable Java client, and IBM Java runtime environment Version 8 on your x86 Linux.

About this task

This task takes you through the minimal setup that is needed to create the IBM MQ Bridge to blockchain configuration file and successfully connect to your IBM Blockchain and IBM MQ networks.

You can use the bridge to connect to blockchain networks that are based on Hyperledger Fabric v1.0 architecture. To use the bridge, you need configuration information from your blockchain network. In each step in this task you can find example configuration details that are based on two differently configured blockchain networks:

- Hyperledger Fabric network that runs in Docker. For more information, see Getting started with Hyperledger Fabric, Writing your first application, and "Example Hyperledger Fabric network credentials file" on page 1446.
- Hyperledger Fabric network that runs in a Kubernetes cluster in IBM Cloud (formerly Bluemix). For more information, see Develop in a cloud sandbox on IBM Blockchain Platform, and "Example Kubernetes container cluster network configuration file" on page 1449.

For more information on the meaning and options for all the IBM MQ Bridge to blockchain parameters, see the runmqbcb command. You must consider your own security requirements and customize the parameters appropriate to your deployment.

Procedure

1. Run the bridge to create a configuration file. You need the parameters from your blockchain network credentials file and from your z/OS queue manager. Run the bridge script from the `bin` directory of the location where you unpacked the bridge when you moved it from your z/OS environment in task "Configuring IBM MQ Advanced for z/OS VUE for use with blockchain" on page 1598.

   ```
   ./runmqbcb -o config_file_name.cfg
   ```

   As the following example illustrates, the existing values are shown inside the brackets. Press `Enter` to accept existing values, press `Space` then `Enter` to clear values, and type inside the brackets then press `Enter` to add new values. You can separate lists of values (such as peers) by commas, or by entering each value on a new line. A blank line ends the list.

   Note: You cannot edit the existing values. You can keep, replace, or clear them.
2. Enter values for the connection to your z/OS queue manager. Minimum values that are needed for the connection are the queue manager name, the names of the bridge input and identity queues that you defined. For connections to remote queue managers, you also need **`MQ Channel`** and **`MQ Conname`** (host address and port where the queue manager is running). To use TLS for connecting to IBM MQ in step 6 on page 1603, you must use JNDI or CCDT and specify **`MQ CCDT URL`** or **`JNDI implementation class`** and **`JNDI provider URL`** accordingly.

   ```
   Connection to Queue Manager
   ---------------------------
   Queue Manager                  : [z/OS_qmgr_name]
   Bridge Input Queue             : [APPL1.BLOCKCHAIN.INPUT.QUEUE]
   Bridge User Identity Queue     : [SYSTEM.BLOCKCHAIN.IDENTITY.QUEUE]
   MQ Channel                     : [SYSTEM.DEF.SVRCONN]
   MQ Conname                     : [host1.example.com(3714)]
   MQ CCDT URL                    : []
   JNDI implementation class      : []
   JNDI provider URL              : []
   MQ Userid                      : []
   MQ Password                    : []
   ```
3. Enter the login details for the certificate authority for your blockchain network. The default values for your local Hyperledger Fabric and Kubernetes cluster examples are *admin* for **`Userid`** and *adminpw* for **`Enrollment Secret`**. If you changed these values for your blockchain network, ensure that you use the correct values to configure the bridge.

   ```
   Blockchain - User Identification
   --------------------------------
   Blockchain Userid              : []admin
   Enrollment Secret              : []******
   ```
4. Enter the membership service provider id (**`MSPid`**) that governs membership and identity rules for your blockchain network. From your credentials file, provide the **`msp_id`** parameter for the **`Organisation Name`** and **`Organisation MSPId`**. From the "Example Hyperledger Fabric network credentials file" on page 1446, use the **`CORE_PEER_LOCALMSPID`** value from the `peer` section of the file. From the "Example Kubernetes container cluster network configuration file" on page 1449, use the **`mspID`** value.

```
Blockchain - Organisation Identification
----------------------------------------
Organisation Name                  : []Org1MSP
Organisation MSPId                 : []Org1MSP
```

5. Enter your blockchain network server location values: From your "Example Hyperledger Fabric network credentials file" on page 1446, provide the names and server:port locations for certificate authority, peer, and orderer elements.

   ```
   Blockchain server locations
   ---------------------------
   Certificate Authority servers      : [ca.example.com Docker_container_host:7054] (for example ca.example.com localh
   Peer servers                       : [peer0 localhost:7051]
   Orderer servers                    : [orderer0 localhost:7050]
   Peer Event servers                 : [peer0 localhost:7053]
   Location of PEM file for Blockchain certificate : []
   ```

   From your "Example Kubernetes container cluster network configuration file" on page 1449, provide the names and server:port locations for certificate authority, peer, and orderer elements.

   ```
   Blockchain server locations
   ---------------------------
   Certificate Authority servers      : [CA1                 your_blockchain_network_public_ip_address:30000] (for ex
   Peer servers                       : [blockchain-org1peer1 your_blockchain_network_public_ip_address:30110]
   Orderer servers                    : [blockchain-orderer   your_blockchain_network_public_ip_address:31010]
   Peer Event servers                 : [blockchain-org1peer1 your_blockchain_network_public_ip_address:30111]
   Location of PEM file for Blockchain certificate : []
   ```

6. Enter certificate stores values for TLS connections. The bridge acts as an IBM MQ Java client that is connecting to a queue manager, which means that it can be configured to use TLS security to connect securely in the same way as any other IBM MQ Java client. Configuration of TLS connection details is exposed only after you specify JNDI or CCDT information in step 2 on page 1602.

   ```
   Certificate stores for TLS connections
   --------------------------------------
   Personal keystore                  : []
   Keystore password                  : []
   Trusted store for signer certs     : []
   Trusted store password             : []
   Use TLS for MQ connection          : [N]
   Timeout for Blockchain operations  : [12]
   ```

7. Enter the location for the log file for the IBM MQ Bridge to blockchain. You must specify the log file name and location, in the configuration file or on the command line.

   ```
   Behavior of bridge program
   --------------------------
   Runtime logfile for copy of stdout/stderr : [/var/mqm/errors/runmqbcb.log]
   Done.
   ```

Results

You created the configuration file that the IBM MQ Bridge to blockchain uses to connect to your IBM Blockchain network and to your IBM MQ z/OS queue manager.

What to do next

Work through the steps for "Running the IBM MQ Bridge to blockchain" on page 1605

Related information:
runmqbcb (run IBM MQ Bridge to blockchain)

Security for queues in use with the IBM MQ Bridge to blockchain

z/OS V 9.0.3

Considerations for setting up security for z/OS queues that are defined for use with the IBM MQ Bridge to blockchain.

The following examples show RACF profiles that illustrate one way of securing the queues for the IBM MQ Bridge to blockchain.

RESLEVEL

The IBM MQ Bridge to blockchain connects through a **SVRCONN** channel to the **CHINIT**. We assume that specific security checking is required on the effective z/OS user ID used by the bridge user. This means we need to ensure that user IDs are checked for **CHINIT** tasks. Authority on the **RESLEVEL** profile determines whether just one user ID (the channel user ID) is checked, or two user IDs (both the **channel** user ID AND the **CHINIT** user ID) are checked. For example:

- This code grants **READ** authority to **CHINIT** in the **RESLEVEL** profile. Therefore only the **channel** user IDs will be checked.
  ```
  PERMIT RESLEVEL CLASS(MQADMIN) ID(CHINIT) ACCESS(READ)
  ```
- This code grants **CHINIT** no authority in the **RESLEVEL** profile. Therefore two user IDs are checked, and additional permissions must be granted to the **CHINIT** user ID.
  ```
  PERMIT RESLEVEL CLASS(MQADMIN) ID(CHINIT) ACCESS(NONE)
  ```

In the next section, the lines of code that grant additional permissions are highlighted.

For more information, see Client MQI requests.

Queue resource authorities

Lock down the identity queue and permit the bridge ID to use it for input and output

```
RDEFINE MQQUEUE SYSTEM.BLOCKCHAIN.IDENTITY.QUEUE UACC(NONE)
PERMIT SYSTEM.BLOCKCHAIN.IDENTITY.QUEUE CLASS(MQQUEUE) ID(MQBBCART) ACCESS(UPDATE)
PERMIT SYSTEM.BLOCKCHAIN.IDENTITY.QUEUE CLASS(MQQUEUE) ID(CHINIT) ACCESS(UPDATE)
```

Bridge ID can open queue for input

```
DEF QL(CARTAX.BLOCKCHAIN.INPUT.QUEUE) LIKE(SYSTEM.BLOCKCHAIN.INPUT.QUEUE)
RDEFINE MQQUEUE CARTAX.BLOCKCHAIN.INPUT.QUEUE UACC(NONE)
PERMIT CARTAX.BLOCKCHAIN.INPUT.QUEUE CLASS(MQQUEUE) ID(MQBBCART) ACCESS(UPDATE)
PERMIT APPL1.BLOCKCHAIN.INPUT.QUEUE CLASS(MQQUEUE) ID(CHINIT) ACCESS(UPDATE)
```

Application IDs in APPCART group can open request queue for output

```
PERMIT CARTAX.BLOCKCHAIN.INPUT.QUEUE CLASS(MQQUEUE) ID(APPCART) ACCESS(UPDATE)
```

Profile to cover application reply queues

```
RDEFINE MQQUEUE CARTAX.APP.REPLY.** UACC(NONE)
```

Application IDs in APPCART group can open reply queue for input

```
RDEFINE MQADMIN CONTEXT.CARTAX.APP.REPLY.** UACC(NONE)
PERMIT CARTAX.APP.REPLY.** CLASS(MQQUEUE) ID(APPCART) ACCESS(UPDATE)
```

Bridge ID can open reply queue for output and put with **set_identity_context**

```
PERMIT CARTAX.APP.REPLY.** CLASS(MQQUEUE) ID(MQBBCART) ACCESS(UPDATE)
PERMIT CONTEXT.CARTAX.APP.REPLY.** CLASS(MQADMIN) ID(MQBBCART) ACCESS(UPDATE)
PERMIT CARTAX.APP.REPLY.** CLASS(MQQUEUE) ID(CHINIT) ACCESS(UPDATE)
PERMIT CONTEXT.CARTAX.APP.REPLY.** CLASS(MQADMIN) ID(CHINIT) ACCESS(UPDATE)
```

Related tasks:

"Running the IBM MQ Bridge to blockchain client sample" on page 1609
You can use the JMS client sample that is provided with the IBM MQ Bridge to blockchain, to put a message on the input queue that the blockchain bridge is checking and see the reply that is received.

Related information:

Profiles for queue security

API-resource security access quick reference

Running the IBM MQ Bridge to blockchain

z/OS Linux V 9.0.3 MQ Adv. VUE

Run the IBM MQ Bridge to blockchain to connect to IBM Blockchain and IBM MQ. When connected, the bridge is ready to process query messages, to send them to your blockchain network, and to receive and process the replies.

About this task

Use the configuration file that you created in the previous task, to run the IBM MQ Bridge to blockchain.

Procedure

1. Start the z/OS queue manager that you want to use with the bridge.
2. Start the IBM MQ Bridge to blockchain to connect to your blockchain network and your z/OS queue manager. Run the bridge script from the `bin` directory of the location where you unpacked the bridge when you moved it from your z/OS environment in task "Configuring IBM MQ Advanced for z/OS VUE for use with blockchain" on page 1598.

   ```
   ./runmqbcb -f /config_file_location/config_file_name.cfg -r /log_file_location/logFile.log
   ```

   When the bridge is connected, output similar to the following is returned:

   ```
   Fri Oct 06 06:32:11 PDT 2017 IBM MQ Bridge to Blockchain
   5724-H72 (C) Copyright IBM Corp. 2017

   Fri Oct 06 06:32:17 PDT 2017 Ready to process input messages.
   ```
3. Optional: Troubleshoot connections to your z/OS queue manager and to your blockchain network, if the messages that are returned after you run the bridge indicate that a connection was not successful.
   a. Issue the command in debug mode with the debug option 1.

      ```
      ./runmqbcb -f /config_file_location/config_file_name.cfg -r /log_file_location/logFile.log -d 1
      ```

      The bridge steps through the connection set up and shows the processing messages in terse mode.
   b. Issue the command in debug mode with the debug option 2.

      ```
      ./runmqbcb -f /config_file_location/config_file_name.cfg -r /log_file_location/logFile.log -d 2
      ```

      The bridge steps through the connection set up and shows the processing messages in verbose mode. Full output is written to your log file.

Results

You have started the IBM MQ Bridge to blockchain and connected to your queue manager and blockchain network.

What to do next

- Follow the steps in "Running the IBM MQ Bridge to blockchain client sample" on page 1609 to format and send a query or update message to your blockchain network.
- Use the *MQBCB_EXTRA_JAVA_OPTIONS* variable to pass in JVM properties, for example to enable IBM MQ tracing. For more information, see Tracing the IBM MQ Bridge to blockchain.

Message formats for the IBM MQ Bridge to blockchain: z/OS Linux V 9.0.3 MQ Adv. VUE

Information on formatting of the messages that are sent and received by the IBM MQ Bridge to blockchain.

An application requests that the IBM MQ Bridge to blockchain performs a query or update of information that is held on the blockchain. The application does this by placing a request message on the bridge request queue. The results of the query or the update are formatted by the bridge into a reply message. The bridge uses information that is contained in the **ReplyToQ** and **ReplyToQMgr** fields from the MQMD of the request message as the destination for the reply message.

The messages that are consumed and produced by the bridge are text (MQSTR) messages in JSON format. The input message is a simple JSON and programs can use string concatenation to generate it. All the fields except **args** are required, the argument list for that field requires knowledge of the functions of the stored chaincode.

Request Message Format

Input message format:

```
{ "function": functionName,
    "channel" : chainName,
    "chaincodeName" : codeName,
    "args" : [ argument list]
}
```

For the local hyperledger network example with the working Fabcar sample.

- To use the query message that calls the `queryAllCars` function in the fabcar chaincode that returns a list of JSON objects that represent the car details that are held in the blockchain, format the message as follows:

  ```
  { "function": "queryAllCars",
      "channel":"mychannel",
      "chaincodeName": "fabcar",
      "args":[]
  }
  ```

  Example reply:

  ```
  {
     "statusCode": 200,
     "statusType": "SUCCESS",
     "message": "OK",
     "data": [
   {"Record":{"owner":"Tomoko","colour":"blue","model":"Prius","make":"Toyota"},"Key":"CAR0"},
   {"Record":{"owner":"Brad","colour":"red","model":"Mustang","make":"Ford"},"Key":"CAR1"},
   {"Record":{"owner":"Jin Soo","colour":"green","model":"Tucson","make":"Hyundai"},"Key":"CAR2"},
   {"Record":{"owner":"Max","colour":"yellow","model":"Passat","make":"Volkswagen"},"Key":"CAR3"},
   {"Record":{"owner":"Adriana","colour":"black","model":"S","make":"Tesla"},"Key":"CAR4"},
   {"Record":{"owner":"Michel","colour":"purple","model":"205","make":"Peugeot"},"Key":"CAR5"},
   {"Record":{"owner":"Aarav","colour":"white","model":"S22L","make":"Chery"},"Key":"CAR6"},
  ```

```
  {"Record":{"owner":"Pari","colour":"violet","model":"Punto","make":"Fiat"},"Key":"CAR7"},
  {"Record":{"owner":"Valeria","colour":"indigo","model":"Nano","make":"Tata"},"Key":"CAR8"},
  {"Record":{"owner":"Shotaro","colour":"brown","model":"Barina","make":"Holden"},"Key":"CAR9"}
]}
```

The reply message contains all the car records that are currently held in the blockchain.

- To use the update message that calls the `createCar` function in the fabcar example chaincode that creates a new car entry in the blockchain ledger, format the message as follows:

```
{ "function":"createCar",
  "channel":"mychannel",
  "chaincodeName":"fabcar",
  "args":["CAR10", "Ford", "Mustang GT", "Blue", "Bob"]
}
```

Example reply:

```
{
   "statusCode": 200,
   "statusType": "SUCCESS",
   "message": "OK",
   "data": ""
}
```

To check that the new car entry is added to the blockchain, you can use the initial message again that returns all the cars.

For the Kubernetes cluster network example with the working example02 demo.

- To use the query message that calls the `query` function in the example02 chaincode that returns the value for entity *"a"* within the blockchain ledger, format the message as follows:

```
{ "function":"query",
  "channel":"channel1",
  "chaincodeName":"example02",
  "args":["a"]
}
```

Example reply:

```
{
   "statusCode": 200,
   "statusType": "SUCCESS",
   "message": "OK",
   "data": "100"
}
```

- To use the message that calls the invoke function example02 chaincode that decrements the entity that is specified in the first argument and increments the entity that is specified in the second argument by the value that is specified in the third argument, format the message as follows:

```
{ "function":"invoke",
  "channel":"channel1",
  "chaincodeName":"example02",
  "args":["a", "b", "10"]
}
```

The values are as follows:

  – Before: a=100, b=200
  – After: a=90, b=210

Example reply:

```
{
   "statusCode": 200,
   "statusType": "SUCCESS",
   "message": "OK",
   "data": ""
}
```

To check the new values, submit a new message query message to look for values of **"a"** and **"b"**.

Reply Message Format

Response messages have their correlation ID set to the message ID of the inbound message. Any user-defined properties are copied from the input to the output messages. The user ID in the reply is set to the originator's user ID through the set-identity context.

An example of successful processing:

```
{ "data": "500", "message": "OK", "statusCode": 200, "statusType": "SUCCESS" }
```

The response data in this message is whatever is generated from the chaincode response (bytes converted to a UTF-8 string).

All error responses have the same fields, regardless of whether they are generated by the bridge itself, from the calls to blockchain, or from the chaincode invocation. For example:

- Bad channel name

```
{
   "message": "Bad newest block expected status 200 got 404, Chain myUnknownChannel",
   "statusCode": 404,
   "statusType": "FAILURE"
}
```

- Bad JSON input message

```
{
   "message": "Error: Cannot parse message contents.",
   "statusCode": 2110,
   "statusType": "FAILURE"
}
```

- Incorrect parameters to chaincode

```
{
   "message": "Sending proposal to fabric-peer-1a failed because of gRPC failure=Status{code=UNKNOWN, description={\"Erro
   "statusCode": 500,
   "statusType": "FAILURE"
}
```

Applications can tell whether the request succeeded or failed by either looking at the **statusType** string, or from the existence of the data field. When there is an error in processing the input message, and the bridge does not send it to blockchain, the value that is returned from the bridge is an MQRC value, usually **MQRC_FORMAT_ERROR**.

Running the IBM MQ Bridge to blockchain client sample: z/OS Linux V 9.0.3 MQ Adv. VUE

You can use the JMS client sample that is provided with the IBM MQ Bridge to blockchain, to put a message on the input queue that the blockchain bridge is checking and see the reply that is received.

Before you begin

Your IBM MQ Bridge to blockchain is running and is connected to your IBM MQ Advanced queue manager and your blockchain network, and is ready to process input messages.

About this task

Find the JMS sample application in the `samp` directory of the IBM MQ Bridge to blockchain.

Procedure

1. Edit the client sample Java source file. Follow the instructions in the sample to configure it to match your IBM MQ environment and your blockchain network. The following code from the sample defines the JSON request message to send to the bridge:

```
 // Create the JSON request message.
 // Modify "query", "exampleBlockchainChannelName", and "exampleChaincodeName" to
 // match your deployed blockchain chaincode.
 // The "operation" field is optional, but recommended. It should be set to QUERY
 // or UPDATE to match what the chaincode is going to do.

JSONObject inputMsg = new JSONObject();
 inputMsg.put("operation", "QUERY");

inputMsg.put("function", "query");
 inputMsg.put("channel", "exampleBlockchainChannelName");
 inputMsg.put("chaincodeName","exampleChaincodeName");


// Create the JSON arguments for the request message.
 // Modify "a" to match your deployed blockchain chaincode
 // requirements, and add further arguments as necessary

JSONArray myArgs = new JSONArray();
 myArgs.add("a");
 inputMsg.put("args", myArgs);


 TextMessage message = session.createTextMessage(inputMsg.serialize());
 message.setJMSReplyTo(replyToQueue);
```

2. Compile the sample. Point to the IBM MQ client classes and `JSON4j.jar` file that is shipped in the bridge directory.

```
javac -cp $MQ_JAVA_INSTALL_PATH/lib/*:../prereqs/JSON4J.jar SimpleBCBClient.java
```

3. Run the compiled class.

```
java -cp $MQ_JAVA_INSTALL_PATH/lib/*:../prereqs/JSON4J.jar:. SimpleBCBClient
```

```
Starting Simple MQ Blockchain Bridge Client
Created the message. Starting the connection
Sent message:

 JMSMessage class: jms_text
  JMSType:          null
  JMSDeliveryMode:  2
  JMSDeliveryDelay: 0
  JMSDeliveryTime:  1508427559117
  JMSExpiration:    0
  JMSPriority:      4
  JMSMessageID:     ID:414d5120424342514d202020202020209063e859ea36aa24
  JMSTimestamp:     1508427559117
  JMSCorrelationID: null
  JMSDestination:   queue:///APPL1.BLOCKCHAIN.INPUT.QUEUE
  JMSReplyTo:       queue:///APPL1.BLOCKCHAIN.REPLY.QUEUE
  JMSRedelivered:   false
    JMSXAppID: java
   JMSXDeliveryCount: 0
    JMSXUserID: USER1
   JMS_IBM_PutApplType: 6
    JMS_IBM_PutDate: 20171019
    JMS_IBM_PutTime: 15391912
{"args":["a"],"function":"query","channel":"exampleBlockchainChannelName","operation":"QUERY","chaincodeName":"exampleCh
```

Response message:

```
  JMSMessage class: jms_text
  JMSType:          null
  JMSDeliveryMode:  1
  JMSDeliveryDelay: 0
  JMSDeliveryTime:  0
  JMSExpiration:    0
  JMSPriority:      4
  JMSMessageID:     ID:c3e2d840e2e2f0f84040404040404040d2afa27229838af2
  JMSTimestamp:     1497439784000
  JMSCorrelationID: ID:414d5120424342514d202020202020209063e859ea36aa24  *(JMSMessageID of the input message)
  JMSDestination:   null
  JMSReplyTo:       null
  JMSRedelivered:   false
    JMSXAppID: java
    JMSXDeliveryCount: 1
    JMSXUserID: USER1
    JMS_IBM_Character_Set: UTF-8
    JMS_IBM_Encoding: 273
    JMS_IBM_Format: MQSTR
    JMS_IBM_MsgType: 8
    JMS_IBM_PutApplType: 2
    JMS_IBM_PutDate: 20171019
    JMS_IBM_PutTime: 15392014
{
   "data": "20",
   "message": "OK",
   "statusCode": 200,
   "statusType": "SUCCESS"
}
Response text:
{
   "data": "20",
   "message": "OK",
   "statusCode": 200,
   "statusType": "SUCCESS"
}
SUCCESS
```

If the client receives a timeout error waiting for the response, check that the bridge is running.

Index

Notices

Programming interface information

Programming interface information, if provided, is intended to help you create application software for use with this program.

This book contains information on intended programming interfaces that allow the customer to write programs to obtain the services of WebSphere MQ.

However, this information may also contain diagnosis, modification, and tuning information. Diagnosis, modification and tuning information is provided to help you debug your application software.

Important: Do not use this diagnosis, modification, and tuning information as a programming interface because it is subject to change.

Trademarks

Sending your comments to IBM

We appreciate your input on this publication. Feel free to comment on the clarity, accuracy, and completeness of the information or give us any other feedback that you might have.

Use one of the following methods to send us your comments:
- Send an email to ibmkc@us.ibm.com
- Use the form on the web here: www.ibm.com/software/data/rcf/

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

Include the following information:
- Your name and address
- Your email address
- Your telephone or fax number
- The publication title and order number
- The topic and page number related to your comment
- The text of your comment

IBM or any other organizations will only use the personal information that you supply to contact you about the issues that you submit.

Thank you for your participation.