9.0

IBM MQ Configuration Reference

Note

Before using this information and the product it supports, read the information in “Notices” on page 181.

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

Configuration reference

Use the reference information in this section to help you configure IBM MQ.

The configuration reference information is provided in the following subtopics:

Related tasks

Configuring

z/OS Configuring z/OS

Example IBM MQ configuration for all platforms

The configuration examples describe tasks performed to establish a working IBM MQ network. The tasks are to establish IBM MQ sender and receiver channels to enable bidirectional message flow between the platforms over all supported protocols.

To use channel types other than sender-receiver, see the DEFINE CHANNEL command.

Figure 1 on page 5 is a conceptual representation of a single channel and the IBM MQ objects associated with it.

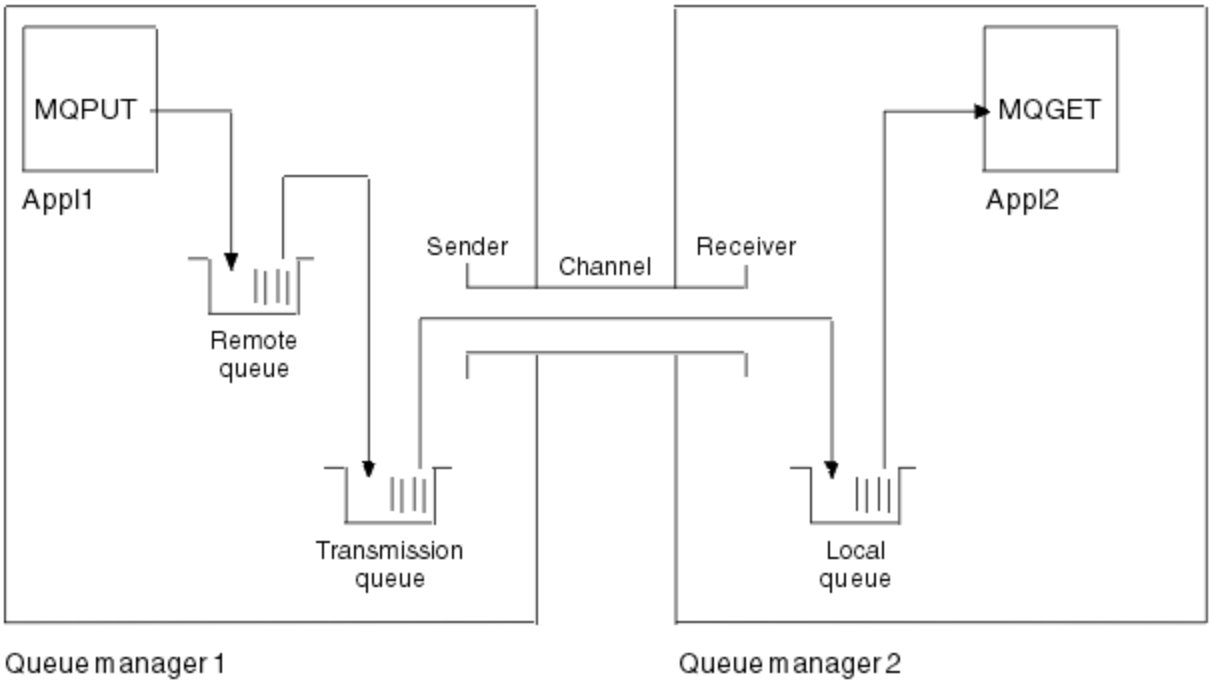

Figure 1. IBM MQ channel to be set up in the example configuration

This example is a simple one, intended to introduce only the basic elements of the IBM MQ network. It does not demonstrate the use of triggering which is described in Triggering channels.

The objects in this network are:

- A remote queue
- A transmission queue
- A local queue
- A sender channel
- A receiver channel

Appl1 and Appl2 are both application programs; Appl1 is putting messages and Appl2 is receiving them.

Appl1 puts messages to a remote queue. The definition for this remote queue specifies the name of a target queue manager, a local queue on that queue manager, and a transmission queue on this local queue manager.

When the queue manager receives the request from Appl1 to put a message to the remote queue, the queue manager determines from the queue definition that the destination is remote. It therefore puts the message, along with a transmission header, straight onto the transmission queue specified in the definition. The message remains on the transmission queue until the channel becomes available, which might happen immediately.

A sender channel has in its definition a reference to one, and one only, transmission queue. When a channel is started, and at other times during its normal operation, it looks at this transmission queue and send any messages on it to the target system. The message has in its transmission header details of the destination queue and queue manager.

The intercommunication examples describe in detail the creation of each of the preceding objects described, for various platform combinations.

On the target queue manager, definitions are required for the local queue and the receiver side of the channel. These objects operate independently of each other and so can be created in any sequence.

On the local queue manager, definitions are required for the remote queue, the transmission queue, and the sender side of the channel. Since both the remote queue definition and the channel definition refer to the transmission queue name, it is advisable to create the transmission queue first.

Network infrastructure in the example

The configuration examples assume that particular network infrastructures are in place for particular platforms:

- z/OS z/OS communicates by using a 3745 network controller (or equivalent) that is attached to a token ring
- Solaris Solaris is on an adjacent local area network (LAN) also attached to a 3745 network controller (or equivalent)
- All other platforms are connected to a token-ring network

It is also assumed that, for SNA, all the required definitions in VTAM and network control program (NCP) are in place and activated for the LAN-attached platforms to communicate over the wide area network (WAN).

Similarly, for TCP, it is assumed that name server function is available, either by using a domain name server or by using locally held tables (for example a host file).

Communications software in the example

Working configurations are given in the examples for the following network software products:

- SNA
  - IBM Personal Communications for Windows 5.9
  - IBM Communications Server for AIX® 6.3
  - Hewlett-Packard SNAplus2
  - IBM i
  - Data Connection SNAP-IX Version 7 or later
  - OS/390® 2.4
- TCP
  - Microsoft Windows
  - AIX 4.1.4

  - HP-UX 10.2 or later
  - Sun Solaris 2.4 or later
  - IBM i
  - TCP for z/OS
  - HP Tru64 UNIX
- NetBIOS
- SPX

Related tasks

Configuring distributed queuing

Setting up communications with other queue managers on z/OS

How to use the communication examples

The example-configurations describe the tasks that are carried out on a single platform to set up communication to another of the platforms. Then they describe the tasks to establish a working channel to that platform.

Wherever possible, the intention is to make the information as generic as possible. Thus, to connect any two queue managers on different platforms, you need to refer to only the relevant two sections. Any deviations or special cases are highlighted as such. You can also connect two queue managers running on the same platform (on different machines or on the same machine). In this case, all the information can be derived from the one section.

ULW On UNIX, Linux®, and Windows, before you begin to follow the instructions for your platform you must set various environment variables. Do this by entering one of the following commands:

- Windows On Windows:

  ```
  MQ_INSTALLATION_PATH/bin/setmqenv
  ```

  where *MQ_INSTALLATION_PATH* refers to the location where IBM MQ is installed.

- Linux UNIX On UNIX and Linux systems:

  ```
  . MQ_INSTALLATION_PATH/bin/setmqenv
  ```

  where *MQ_INSTALLATION_PATH* refers to the location where IBM MQ is installed. This command sets the environment variables for the shell you are currently working in. If you open another shell, you must enter the command again.

There are examples in which you can find the parameters used in the sample configurations. There is a short description of each parameter and some guidance on where to find the equivalent values in your system. When you have a set of values of your own, make sure that you use those values when working through the examples in this section.

The examples do not cover how to set up communications where clustering is being used. For information about setting up communications while using clustering, see Configuring a queue manager cluster. The communication configuration values given here still apply.

There are example configurations for the following platforms:

- AIX "Example IBM MQ configuration on AIX" on page 9
- HP-UX "Example IBM MQ configuration for HP-UX" on page 15
- IBM i "Example MQ configuration for IBM i" on page 21
- Linux "Example MQ configuration for Linux" on page 37

- Solaris "Example MQ configuration for Solaris" on page 43
- Windows "Example IBM MQ configuration for Windows" on page 49
- z/OS "Example MQ configuration for z/OS" on page 56
- z/OS "Example MQ configuration for z/OS using QSGs" on page 61
- z/OS "Example MQ configuration for z/OS using intra-group queuing" on page 68

IT responsibilities

To understand the terminology used in the examples, consider the following guidelines as a starting point.

- System administrator: The person (or group of people) who installs and configures the software for a specific platform.
- Network administrator: The person who controls LAN connectivity, LAN address assignments, network naming conventions, and other network tasks. This person can be in a separate group or can be part of the system administration group.

  In most z/OS installations, there is a group responsible for updating the ACF/VTAM, ACF/NCP, and TCP/IP software to support the network configuration. The people in this group are the main source of information needed when connecting any IBM MQ platform to IBM MQ for z/OS. They can also influence or mandate network naming conventions on LANs and you must verify their span of control before creating your definitions.
- A specific type of administrator, for example CICS® administrator, is indicated in cases where we can more clearly describe the responsibilities of the person.

The example-configuration sections do not attempt to indicate who is responsible for and able to set each parameter. In general, several different people might be involved.

Related concepts

"Example IBM MQ configuration for all platforms" on page 5
The configuration examples describe tasks performed to establish a working IBM MQ network. The tasks are to establish IBM MQ sender and receiver channels to enable bidirectional message flow between the platforms over all supported protocols.

Related reference

setmqenv

Multi Multiple thread support - pipelining

You can optionally allow a message channel agent (MCA) to transfer messages using multiple threads. This process, called *pipelining*, enables the MCA to transfer messages more efficiently, with fewer wait states, which improves channel performance. Each MCA is limited to a maximum of two threads.

You control pipelining with the *PipeLineLength* parameter in the qm.ini file. This parameter is added to the CHANNELS stanza:

PipeLineLength= 1 | *number*
: This attribute specifies the maximum number of concurrent threads a channel uses. The default is 1. Any value greater than 1 is treated as 2.

Note: Pipelining is effective only for TCP/IP channels.

When you use pipelining, the queue managers at both ends of the channel must be configured to have a *PipeLineLength* greater than 1.

Channel exit considerations

Pipelining can cause some exit programs to fail, because:

- Exits might not be called serially.
- Exits might be called alternately from different threads.

Check the design of your exit programs before you use pipelining:

- Exits must be reentrant at all stages of their execution.
- When you use MQI calls, remember that you cannot use the same MQI handle when the exit is invoked from different threads.

Consider a message exit that opens a queue and uses its handle for MQPUT calls on all subsequent invocations of the exit. This fails in pipelining mode because the exit is called from different threads. To avoid this failure, keep a queue handle for each thread and check the thread identifier each time the exit is invoked.

AIX Example IBM MQ configuration on AIX

This section gives an example of how to set up communication links from IBM MQ for AIX to IBM MQ products.

The following platforms are covered in the examples:

- Windows Windows
- UNIX HP Tru64 UNIX
- HP-UX HP-UX
- Solaris Solaris
- Linux Linux
- IBM i IBM i
- z/OS z/OS
- VSE/ESA

See "Example IBM MQ configuration for all platforms" on page 5 for background information about this section and how to use it.

AIX Establishing an LU 6.2 connection

Describes the parameters needed for an LU 6.2 connection.

For the latest information about configuring SNA over TCP/IP, refer to the following online IBM documentation: Communications Server for AIX .

AIX Establishing a TCP connection

The listener must be started explicitly before any channels are started. It enables receiving channels to start automatically in response to a request from an inbound sending channel.

The IBM MQ command used to start the IBM MQ for TCP listener is:

```
runmqlsr -t tcp
```

Alternatively, if you want to use the UNIX supplied TCP/IP listener, complete the following steps:

1. Edit the file /etc/services.

   Note: To edit the /etc/services file, you must be logged in as a superuser or root. If you do not have the following line in that file, add it as shown:

```
MQSeries       1414/tcp      # MQSeries channel listener
```

2. Edit the file /etc/inetd.conf. If you do not have the following line in that file, add it as shown, replacing *MQ_INSTALLATION_PATH* with the high-level directory in which IBM MQ is installed:

```
MQSeries stream tcp nowait root MQ_INSTALLATION_PATH/bin/amqcrsta amqcrsta
[-m queue.manager.name]
```

3. Enter the command `refresh -s inetd`.

Note: You must add **root** to the mqm group. You need not have the primary group set to mqm. As long as mqm is in the set of groups, you can use the commands. If you are running only applications that use the queue manager you do not need mqm group authority.

What next?

The connection is now established. You are ready to complete the configuration. Go to "IBM MQ for AIX configuration" on page 10.

AIX IBM MQ for AIX configuration

Defining channels to complete the configuration.

Note:

1. Before beginning the installation process ensure that you have first created the *mqm* user and group, and set the password.
2. If installation fails as a result of insufficient space in the file system you can increase the size as follows, using the command `smit C sna`. (Use `df` to display the status of the file system. This indicates the logical volume that is full.)

```
-- Physical and Logical Storage
  -- File Systems
    -- Add / Change / Show / Delete File Systems
      -- Journaled File Systems
        -- Change/Show Characteristics of a Journaled File System
```

3. Start any channel using the command:

```
runmqchl -c channel.name
```

4. Sample programs are installed in *MQ_INSTALLATION_PATH*/samp, where *MQ_INSTALLATION_PATH* represents the high-level directory in which IBM MQ is installed.
5. Error logs are stored in /var/mqm/qmgrs/ *qmgrname* /errors.
6. On AIX, you can start a trace of the IBM MQ components by using standard IBM MQ trace commands, or using AIX system trace. See Using trace for more information about IBM MQ Trace and AIX system trace.
7. When you are using the command interpreter **runmqsc** to enter administration commands, a + at the end of a line indicates that the next line is a continuation. Ensure that there is a space between the last parameter and the continuation character.

Basic configuration

1. Create the queue manager from the AIX command line using the command:

```
crtmqm -u dlqname -q aix
```

where:

aix
Is the name of the queue manager

-q
Indicates that this is to become the default queue manager

-u *dlqname*
Specifies the name of the undeliverable message queue

This command creates a queue manager and a set of default objects.

2. Start the queue manager from the AIX command line using the command:

```
strmqm aix
```

where *aix* is the name given to the queue manager when it was created.

3. Start **runmqsc** from the AIX command line and use it to create the undeliverable message queue by entering the command:

```
def ql (dlqname)
```

where *dlqname* is the name given to the undeliverable message queue when the queue manager was created.

AIX *Channel configuration for AIX*

Includes information about configuring a queue manager for a given channel and platform.

The following section details the configuration to be performed on the AIX queue manager to implement the channel described in "Example IBM MQ configuration for all platforms" on page 5.

In each case the MQSC command is shown. Either start **runmqsc** from an AIX command line and enter each command in turn, or build the commands into a command file.

Examples are given for connecting IBM MQ for AIX and IBM MQ for Windows. To connect to IBM MQ on another platform use the appropriate set of values from the table in place of those for Windows.

Note: The words in **bold** are suggested values and reflect the names of IBM MQ objects used throughout these examples. You can change them in your product installation but, if you do, make sure that you use your own values when working through the examples in this section

Table 1. Configuration examples for IBM MQ for AIX

ID	Parameter Name	Reference	Example Used
Definition for local node			
A	Queue Manager Name		**AIX**
B	Local queue name		**AIX.LOCALQ**
Windows Windows ***Connection to IBM MQ for Windows*** The values in this section of the table must match those used in "Channel configuration for Windows" on page 52, as indicated.			
C	Remote queue manager name	A	**WINNT**
D	Remote queue name		**WINNT.REMOTEQ**
E	Queue name at remote system	B	**WINNT.LOCALQ**
F	Transmission queue name		**WINNT**
G	Sender (SNA) channel name		**AIX.WINNT.SNA**

Table 1. Configuration examples for IBM MQ for AIX (continued)			
ID	**Parameter Name**	**Reference**	**Example Used**
H	Sender (TCP/IP) channel name		**AIX.WINNT.TCP**
I	Receiver (SNA) channel name	G	**WINNT.AIX.SNA**
J	Receiver (TCP) channel name	H	**WINNT.AIX.TCP**
HP-UX HP-UX ***Connection to IBM MQ for HP-UX*** The values in this section of the table must match those used in "Channel configuration for HP-UX" on page 17, as indicated.			
C	Remote queue manager name	A	**HPUX**
D	Remote queue name		**HPUX.REMOTEQ**
E	Queue name at remote system	B	**HPUX.LOCALQ**
F	Transmission queue name		**HPUX**
G	Sender (SNA) channel name		**AIX.HPUX.SNA**
H	Sender (TCP) channel name		**AIX.HPUX.TCP**
I	Receiver (SNA) channel name	G	**HPUX.AIX.SNA**
J	Receiver (TCP) channel name	H	**HPUX.AIX.TCP**
Solaris Solaris ***Connection to IBM MQ for Solaris*** The values in this section of the table must match those used in "Channel configuration for Solaris" on page 45, as indicated.			
C	Remote queue manager name	A	**SOLARIS**
D	Remote queue name		**SOLARIS.REMOTEQ**
E	Queue name at remote system	B	**SOLARIS.LOCALQ**
F	Transmission queue name		**SOLARIS**
G	Sender (SNA) channel name		**AIX.SOLARIS.SNA**
H	Sender (TCP/IP) channel name		**AIX.SOLARIS.TCP**
I	Receiver (SNA) channel name	G	**SOLARIS.AIX.SNA**
J	Receiver (TCP/IP) channel name	H	**SOLARIS.AIX.TCP**
Linux Linux ***Connection to IBM MQ for Linux*** The values in this section of the table must match those used in "Channel configuration for Linux" on page 40, as indicated.			
C	Remote queue manager name	A	**LINUX**
D	Remote queue name		**LINUX.REMOTEQ**
E	Queue name at remote system	B	**LINUX.LOCALQ**
F	Transmission queue name		**LINUX**
G	Sender (SNA) channel name		**AIX.LINUX.SNA**

Table 1. Configuration examples for IBM MQ for AIX (continued)			
ID	**Parameter Name**	**Reference**	**Example Used**
H	Sender (TCP/IP) channel name		**AIX.LINUX.TCP**
I	Receiver (SNA) channel name	G	**LINUX.AIX.SNA**
J	Receiver (TCP/IP) channel name	H	**LINUX.AIX.TCP**
IBM i IBM i ***Connection to IBM MQ for IBM i*** The values in this section of the table must match those used in "Channel configuration for IBM i" on page 33, as indicated.			
C	Remote queue manager name	A	**AS400**
D	Remote queue name		**AS400.REMOTEQ**
E	Queue name at remote system	B	**AS400.LOCALQ**
F	Transmission queue name		**AS400**
G	Sender (SNA) channel name		**AIX.AS400.SNA**
H	Sender (TCP) channel name		**AIX.AS400.TCP**
I	Receiver (SNA) channel name	G	**AS400.AIX.SNA**
J	Receiver (TCP) channel name	H	**AS400.AIX.TCP**
z/OS z/OS ***Connection to IBM MQ for z/OS*** The values in this section of the table must match those used in "Channel configuration for z/OS" on page 57, as indicated.			
C	Remote queue manager name	A	**MVS**
D	Remote queue name		**MVS.REMOTEQ**
E	Queue name at remote system	B	**MVS.LOCALQ**
F	Transmission queue name		**MVS**
G	Sender (SNA) channel name		**AIX.MVS.SNA**
H	Sender (TCP) channel name		**AIX.MVS.TCP**
I	Receiver (SNA) channel name	G	**MVS.AIX.SNA**
J	Receiver (TCP) channel name	H	**MVS.AIX.TCP**
z/OS z/OS ***Connection to IBM MQ for z/OS using queue sharing groups*** The values in this section of the table must match those used in "Shared channel configuration example" on page 66, as indicated.			
C	Remote queue manager name	A	**QSG**
D	Remote queue name		**QSG.REMOTEQ**
E	Queue name at remote system	B	**QSG.SHAREDQ**
F	Transmission queue name		**QSG**
G	Sender (SNA) channel name		**AIX.QSG.SNA**

Table 1. Configuration examples for IBM MQ for AIX (continued)			
ID	**Parameter Name**	**Reference**	**Example Used**
H	Sender (TCP) channel name		**AIX.QSG.TCP**
I	Receiver (SNA) channel name	G	**QSG.AIX.SNA**
J	Receiver (TCP) channel name	H	**QSG.AIX.TCP**

AIX *IBM MQ for AIX sender-channel definitions using SNA*

Example commands.

```
def ql (WINNT) +                                F
    usage(xmitq) +
    replace

def qr (WINNT.REMOTEQ) +                        D
    rname(WINNT.LOCALQ) +                       E
    rqmname(WINNT) +                            C
    xmitq(WINNT) +                              F
    replace

def chl (AIX.WINNT.SNA) chltype(sdr) +          G
    trptype(lu62) +
    conname('WINNTCPIC') +                      17
    xmitq(WINNT) +                              F
    replace
```

AIX *IBM MQ for AIX receiver-channel definitions using SNA*

Example commands.

```
def ql (AIX.LOCALQ) replace                   B

def chl (WINNT.AIX.SNA) chltype(rcvr) +       I
    trptype(lu62) +
    replace
```

AIX *IBM MQ for AIX TPN setup*

Alternative ways of ensuring that SNA receiver channels activate correctly when a sender channel initiates a conversation.

During the AIX Communications Server configuration process, an LU 6.2 TPN profile was created, which contained the full path to a TP executable program. In the example, the file was called u/interops/AIX.crs6a. You can choose a name, but consider including the name of your queue manager in it. The contents of the executable file must be:

```
#!/bin/sh
MQ_INSTALLATION_PATH/bin/amqcrs6a -m aix
```

where *aix* is the queue manager name (A) and *MQ_INSTALLATION_PATH* is the high-level directory in which IBM MQ is installed. After creating this file, enable it for execution by running the command:

```
chmod 755 /u/interops/AIX.crs6a
```

As an alternative to creating an executable file, you can specify the path on the Add LU 6.2 TPN Profile panel, using command-line parameters.

Specifying a path in one of these two ways ensures that SNA receiver channels activate correctly when a sender channel initiates a conversation.

AIX *IBM MQ for AIX sender-channel definitions using TCP*

Example commands.

```
def ql (WINNT) +                                F
    usage(xmitq) +
    replace

def qr (WINNT.REMOTEQ) +                        D
    rname(WINNT.LOCALQ) +                       E
    rqmname(WINNT) +                            C
    xmitq(WINNT) +                              F
    replace

def chl (AIX.WINNT.TCP) chltype(sdr) +          H
    trptype(tcp) +
    conname(remote_tcpip_hostname) +
    xmitq(WINNT) +                              F
    replace
```

AIX *IBM MQ for AIX receiver-channel definitions using TCP*

Example commands.

```
def ql (AIX.LOCALQ) replace                     B

def chl (WINNT.AIX.TCP) chltype(rcvr) +         J
    trptype(tcp) +
    replace
```

HP-UX Example IBM MQ configuration for HP-UX

This section gives an example of how to set up communication links from IBM MQ for HP-UX to IBM MQ products.

The following platforms are included:

- Windows
- AIX
- HP Tru64 UNIX
- Solaris
- Linux
- IBM i
- z/OS
- VSE/ESA

See "Example IBM MQ configuration for all platforms" on page 5 for background information about this section and how to use it.

HP-UX Establishing an LU 6.2 connection

Describes the parameters needed for an LU 6.2 connection

For the latest information about configuring SNA over TCP/IP, refer to the following online IBM documentation: Communications Server, and the following online HP documentation: HP-UX SNAplus2 Installation Guide.

HP-UX Establishing a TCP connection

Alternative ways of establishing a connection and next steps.

The listener must be started explicitly before any channels are started. It enables receiving channels to start automatically in response to a request from an inbound sending channel.

Alternatively, if you want to use the UNIX supplied TCP/IP listener, complete the following steps:

1. Edit the file /etc/services.

   Note: To edit the /etc/services file, you must be logged in as a superuser or root. If you do not have the following line in that file, add it as shown:

   ```
   MQSeries       1414/tcp      # MQSeries channel listener
   ```

2. Edit the file /etc/inetd.conf. If you do not have the following line in that file, add it as shown, replacing *MQ_INSTALLATION_PATH* with the high-level directory in which IBM MQ is installed.

   ```
   MQSeries stream tcp nowait root MQ_INSTALLATION_PATH/bin/amqcrsta amqcrsta
   [-m queue.manager.name]
   ```

3. Find the process ID of the inetd with the command:

   ```
   ps -ef | grep inetd
   ```

4. Run the command:

   ```
   kill -1 inetd processid
   ```

Note: You must add **root** to the mqm group. You do not need not have the primary group set to mqm. As long as mqm is in the set of groups, you can use the commands. If you are running only applications that use the queue manager you do not need to have mqm group authority.

What next?

The connection is now established. You are ready to complete the configuration. Go to "IBM MQ for HP-UX configuration" on page 16.

HP-UX IBM MQ for HP-UX configuration

Describes defining the channels to complete the configuration.

Before beginning the installation process ensure that you have first created the *mqm* user and group, and set the password.

Start any channel using the command:

```
runmqchl -c channel.name
```

Note:

1. Sample programs are installed in *MQ_INSTALLATION_PATH*/samp, where *MQ_INSTALLATION_PATH* represents the high-level directory in which IBM MQ is installed.
2. Error logs are stored in /var/mqm/qmgrs/ *qmgrname* /errors.
3. When you are using the command interpreter **runmqsc** to enter administration commands, a + at the end of a line indicates that the next line is a continuation. Ensure that there is a space between the last parameter and the continuation character.

Basic configuration

1. Create the queue manager from the UNIX prompt using the command:

   ```
   crtmqm -u dlqname -q hpux
   ```

where:

hpux
Is the name of the queue manager

-q
Indicates that this is to become the default queue manager

-u *dlqname*
Specifies the name of the undeliverable message queue

This command creates a queue manager and a set of default objects. It sets the DEADQ attribute of the queue manager but does not create the undeliverable message queue.

2. Start the queue manager from the UNIX prompt using the command:

```
strmqm hpux
```

where *hpux* is the name given to the queue manager when it was created.

HP-UX *Channel configuration for HP-UX*

Includes information about configuring a queue manager for a given channel and platform.

The following section details the configuration to be performed on the HP-UX queue manager to implement the channel described in "Example IBM MQ configuration for all platforms" on page 5.

In each case the MQSC command is shown. Either start **runmqsc** from a UNIX prompt and enter each command in turn, or build the commands into a command file.

Examples are given for connecting IBM MQ for HP-UX and IBM MQ for Windows. To connect to IBM MQ on another platform use the appropriate set of values from the table in place of those for Windows.

Note: The words in **bold** are user-specified and reflect the names of IBM MQ objects used throughout these examples. If you change the names used here, ensure that you also change the other references made to these objects throughout this section. All others are keywords and should be entered as shown.

Table 2. Configuration worksheet for IBM MQ for HP-UX

ID	Parameter Name	Reference	Example Used	User Value
Definition for local node				
A	Queue Manager Name		**HPUX**	
B	Local queue name		**HPUX.LOCALQ**	
Connection to IBM MQ for Windows The values in this section of the table must match those used in "Channel configuration for Windows" on page 52, as indicated.				
C	Remote queue manager name	A	**WINNT**	
D	Remote queue name		**WINNT.REMOTEQ**	
E	Queue name at remote system	B	**WINNT.LOCALQ**	
F	Transmission queue name		**WINNT**	
G	Sender (SNA) channel name		**HPUX.WINNT.SNA**	
H	Sender (TCP/IP) channel name		**HPUX.WINNT.TCP**	
I	Receiver (SNA) channel name	G	**WINNT.HPUX.SNA**	
J	Receiver (TCP) channel name	H	**WINNT.HPUX.TCP**	

Table 2. Configuration worksheet for IBM MQ for HP-UX (continued)				
ID	**Parameter Name**	**Reference**	**Example Used**	**User Value**
Connection to IBM MQ for AIX The values in this section of the table must match those used in Table 1 on page 11, as indicated.				
C	Remote queue manager name	A	**AIX**	
D	Remote queue name		**AIX.REMOTEQ**	
E	Queue name at remote system	B	**AIX.LOCALQ**	
F	Transmission queue name		**AIX**	
G	Sender (SNA) channel name		**HPUX.AIX.SNA**	
H	Sender (TCP) channel name		**HPUX.AIX.TCP**	
I	Receiver (SNA) channel name	G	**AIX.HPUX.SNA**	
J	Receiver (TCP) channel name	H	**AIX.HPUX.TCP**	
Connection to IBM MQ for HP Tru64 UNIX The values in this section of the table must match those used in your HP Tru64 UNIX system.				
C	Remote queue manager name	A	**DECUX**	
D	Remote queue name		**DECUX.REMOTEQ**	
E	Queue name at remote system	B	**DECUX.LOCALQ**	
F	Transmission queue name		**DECUX**	
H	Sender (TCP) channel name		**DECUX.HPUX.TCP**	
J	Receiver (TCP) channel name	H	**HPUX.DECUX.TCP**	
Connection to IBM MQ for Solaris The values in this section of the table must match those used in Table 6 on page 46, as indicated.				
C	Remote queue manager name	A	**SOLARIS**	
D	Remote queue name		**SOLARIS.REMOTEQ**	
E	Queue name at remote system	B	**SOLARIS.LOCALQ**	
F	Transmission queue name		**SOLARIS**	
G	Sender (SNA) channel name		**HPUX.SOLARIS.SNA**	
H	Sender (TCP/IP) channel name		**HPUX.SOLARIS.TCP**	
I	Receiver (SNA) channel name	G	**SOLARIS.HPUX.SNA**	
J	Receiver (TCP/IP) channel name	H	**SOLARIS.HPUX.TCP**	
Connection to IBM MQ for Linux The values in this section of the table must match those used in Table 5 on page 41, as indicated.				
C	Remote queue manager name	A	**LINUX**	
D	Remote queue name		**LINUX.REMOTEQ**	

Table 2. Configuration worksheet for IBM MQ for HP-UX (continued)				
ID	**Parameter Name**	**Reference**	**Example Used**	**User Value**
E	Queue name at remote system	B	**LINUX.LOCALQ**	
F	Transmission queue name		**LINUX**	
G	Sender (SNA) channel name		**HPUX.LINUX.SNA**	
H	Sender (TCP/IP) channel name		**HPUX.LINUX.TCP**	
I	Receiver (SNA) channel name	G	**LINUX.HPUX.SNA**	
J	Receiver (TCP/IP) channel name	H	**LINUX.HPUX.TCP**	
IBM i ***Connection to IBM MQ for IBM i*** IBM i The values in this section of the table must match those used in Table 4 on page 34, as indicated.				
C	Remote queue manager name	A	**AS400**	
D	Remote queue name		**AS400.REMOTEQ**	
E	Queue name at remote system	B	**AS400.LOCALQ**	
F	Transmission queue name		**AS400**	
G	Sender (SNA) channel name		**HPUX.AS400.SNA**	
H	Sender (TCP/IP) channel name		**HPUX.AS400.TCP**	
I	Receiver (SNA) channel name	G	**AS400.HPUX.SNA**	
IBM i IBM i J	Receiver (TCP) channel name	H	**AS400.HPUX.TCP**	
z/OS ***Connection to IBM MQ for z/OS*** z/OS The values in this section of the table must match those used in Table 8 on page 58, as indicated.				
C	Remote queue manager name	A	**MVS**	
D	Remote queue name		**MVS.REMOTEQ**	
E	Queue name at remote system	B	**MVS.LOCALQ**	
F	Transmission queue name		**MVS**	
G	Sender (SNA) channel name		**HPUX.MVS.SNA**	
H	Sender (TCP) channel name		**HPUX.MVS.TCP**	
I	Receiver (SNA) channel name	G	**MVS.HPUX.SNA**	
z/OS z/OS J	Receiver (TCP) channel name	H	**MVS.HPUX.TCP**	

Table 2. Configuration worksheet for IBM MQ for HP-UX (continued)				
ID	**Parameter Name**	**Reference**	**Example Used**	**User Value**
Connection to MQSeries® for VSE/ESA The values in this section of the table must match those used in your VSE/ESA system.				
C	Remote queue manager name	A	**VSE**	
D	Remote queue name		**VSE.REMOTEQ**	
E	Queue name at remote system	B	**VSE.LOCALQ**	
F	Transmission queue name		**VSE**	
G	Sender channel name		**HPUX.VSE.SNA**	
I	Receiver channel name	G	**VSE.HPUX.SNA**	

HP-UX *IBM MQ for HP-UX sender-channel definitions using SNA*

Example commands.

```
def ql (WINNT) +                                F
    usage(xmitq) +
    replace

def qr (WINNT.REMOTEQ) +                        D
    rname(WINNT.LOCALQ) +                       E
    rqmname(WINNT) +                            C
    xmitq(WINNT) +                              F
    replace

def chl (HPUX.WINNT.SNA) chltype(sdr) +         G
    trptype(lu62) +
    conname('WINNTCPIC') +                      16
    xmitq(WINNT) +                              F
    replace
```

HP-UX *IBM MQ for HP-UX receiver-channel definitions using SNA*

Example commands.

```
def ql (HPUX.LOCALQ) replace                  B

def chl (WINNT.HPUX.SNA) chltype(rcvr) +      I
    trptype(lu62) +
    replace
```

HP-UX *IBM MQ for HP-UX invokable TP setup*

Ensuring that SNA receiver channels activate correctly when a sender channel initiates a conversation.

This is not required for HP SNAplus2 Release 6.

During the HP SNAplus2 configuration process, you created an invokable TP definition, which points to an executable file. In the example, the file was called /users/interops/HPUX.crs6a. You can choose what you call this file, but consider including the name of your queue manager in the name. The contents of the executable file must be:

```
#!/bin/sh
MQ_INSTALLATION_PATH/bin/amqcrs6a -m hpux
```

where *hpux* is the name of your queue manager A and *MQ_INSTALLATION_PATH* is the high-level directory in which IBM MQ is installed.

This ensures that SNA receiver channels activate correctly when a sender channel initiates a conversation.

HP-UX *IBM MQ for HP-UX sender-channel definitions using TCP*

Example commands.

```
def ql (WINNT) +                                F
    usage(xmitq) +
    replace

def qr (WINNT.REMOTEQ) +                        D
    rname(WINNT.LOCALQ) +                       E
    rqmname(WINNT) +                            C
    xmitq(WINNT) +                              F
    replace

def chl (HPUX.WINNT.TCP) chltype(sdr) +         H
    trptype(tcp) +
    conname(remote_tcpip_hostname) +
    xmitq(WINNT) +                              F
    replace
```

HP-UX *IBM MQ for HP-UX receiver-channel definitions using TCP/IP*

Example commands.

```
def ql (HPUX.LOCALQ) replace                  B

def chl (WINNT.HPUX.TCP) chltype(rcvr) +      J
    trptype(tcp) +
    replace
```

IBM i Example MQ configuration for IBM i

This section gives an example of how to set up communication links from IBM MQ for IBM i to IBM MQ products on other platforms.

Other platforms covered are the following platforms:

- Windows
- AIX
- Compaq Tru64 UNIX
- HP-UX
- Solaris
- Linux
- z/OS or MVS
- VSE/ESA

See "Example IBM MQ configuration for all platforms" on page 5 for background information about this section and how to use it.

IBM i Configuration parameters for an LU 6.2 connection

The following worksheet lists all the parameters needed to set up communication from IBM i system to one of the other IBM MQ platforms. The worksheet shows examples of the parameters, which have been tested in a working environment, and leaves space for you to enter your own values.

Use the worksheet in this section to record the values for this configuration. Use the worksheet with the worksheet in the section for the platform to which you are connecting.

Where numbers appear in the Reference column they indicate that the value must match that in the appropriate worksheet elsewhere in this section. The examples that follow in this section refer to the values in the ID column of this table.

The entries in the Parameter Name column are explained in "Explanation of terms" on page 24.

Table 3. Configuration worksheet for SNA on an IBM i system

ID	Parameter Name	Reference	Example Used	User Value
Definition for local node				
1	Local network ID		**NETID**	
2	Local control point name		**AS400PU**	
3	LU name		**AS400LU**	
4	LAN destination address		**10005A5962EF**	
5	Subsystem description		**QCMN**	
6	Line description		**TOKENRINGL**	
7	Resource name		**LIN041**	
8	Local Transaction Program name		**MQSERIES**	
Connection to a Windows system				
9	Network ID	2	**NETID**	
10	Control point name	3	**WINNTCP**	
11	LU name	5	**WINNTLU**	
12	Controller description		**WINNTCP**	
13	Device		**WINNTLU**	
14	Side information		**NTCPIC**	
15	Transaction Program	7	**MQSERIES**	
16	LAN adapter address	9	**08005AA5FAB9**	
17	Mode	17	**#INTER**	
Connection to an AIX system				
9	Network ID	1	**NETID**	
10	Control point name	2	**AIXPU**	
11	LU name	4	**AIXLU**	
12	Controller description		**AIXPU**	
13	Device		**AIXLU**	
14	Side information		**AIXCPIC**	
15	Transaction Program	6	**MQSERIES**	
16	LAN adapter address	8	**123456789012**	
17	Mode	14	**#INTER**	
Connection to an HP-UX system				
9	Network ID	4	**NETID**	

Table 3. Configuration worksheet for SNA on an IBM i system (continued)				
ID	**Parameter Name**	**Reference**	**Example Used**	**User Value**
10	Control point name	2	**HPUXPU**	
11	LU name	5	**HPUXLU**	
12	Controller description		**HPUXPU**	
13	Device		**HPUXLU**	
14	Side information		**HPUXCPIC**	
15	Transaction Program	7	**MQSERIES**	
16	LAN adapter address	8	**100090DC2C7C**	
17	Mode	17	**#INTER**	
Connection to a Solaris system				
9	Network ID	2	**NETID**	
10	Control point name	3	**SOLARPU**	
11	LU name	7	**SOLARLU**	
12	Controller description		**SOLARPU**	
13	Device		**SOLARLU**	
14	Side information		**SOLCPIC**	
15	Transaction Program	8	**MQSERIES**	
16	LAN adapter address	5	**08002071CC8A**	
17	Mode	17	**#INTER**	
Connection to a Linux (x86 platform) system				
9	Network ID	4	**NETID**	
10	Control point name	2	**LINUXPU**	
11	LU name	5	**LINUXLU**	
12	Controller description		**LINUXPU**	
13	Device		**LINUXLU**	
14	Side information		**LXCPIC**	
15	Transaction Program	7	**MQSERIES**	
16	LAN adapter address	8	**08005AC6DF33**	
17	Mode	6	**#INTER**	
Connection to an z/OS system				
9	Network ID	2	**NETID**	
10	Control point name	3	**MVSPU**	
11	LU name	4	**MVSLU**	
12	Controller description		**MVSPU**	
13	Device		**MVSLU**	

Table 3. Configuration worksheet for SNA on an IBM i system (continued)				
ID	**Parameter Name**	**Reference**	**Example Used**	**User Value**
14	Side information		**MVSCPIC**	
15	Transaction Program	7	**MQSERIES**	
16	LAN adapter address	8	**400074511092**	
17	Mode	6	**#INTER**	
Connection to a VSE/ESA system ed				
9	Network ID	1	**NETID**	
10	Control point name	2	**VSEPU**	
11	LU name	3	**VSELU**	
12	Controller description		**VSEPU**	
13	Device		**VSELU**	
14	Side information		**VSECPIC**	
15	Transaction Program	4	**MQ01**	**MQ01**
16	LAN adapter address	5	**400074511092**	
17	Mode		**#INTER**	

IBM i *Explanation of terms*

An explanation of the terms used in the configuration worksheet.

1 2 3
See "How to find network attributes" on page 25 for the details of how to find the configured values.

4 LAN destination address
The hardware address of the IBM i system token-ring adapter. You can find the value using the command DSPLIND *Line description* (6).

5 Subsystem description
This parameter is the name of any IBM i subsystem that is active while using the queue manager. The name QCMN has been used because it is the IBM i communications subsystem.

6 Line description
If this parameter has been specified it is indicated in the Description field of the resource Resource name. See "How to find the value of Resource name" on page 25 for details. If the value is not specified you need to create a line description.

7 Resource name
See "How to find the value of Resource name" on page 25 for details of how to find the configured value.

8 Local Transaction Program name
IBM MQ applications trying to converse with this workstation specify a symbolic name for the program to be run at the receiving end. This name is defined on the channel definition at the sender. For simplicity, wherever possible use a transaction program name of MQSERIES, or in the case of a connection to VSE/ESA, where the length is limited to 4 bytes, use MQTP.

See Settings on the local IBM i system for a remote queue manager platform for more information.

12 Controller description
This parameter is an alias for the Control Point name (or Node name) of the partner system. For convenience, we have used the actual name of the partner in this example.

13 Device
This parameter is an alias for the LU of the partner system. For convenience, we have used the LU name of the partner in this example.

14 Side information
This parameter is the name given to the CPI-C side information profile. You specify your own 8-character name.

IBM i *How to find network attributes*

The local node has been partially configured as part of the IBM i installation. To display the current network attributes enter the command DSPNETA.

If you need to change these values use the command CHGNETA. An IPL might be required to apply your changes.

```
Display Network Attributes
System:  AS400PU
Current system name . . . . . . . . . . . . . . . :  AS400PU
Pending system name . . . . . . . . . . . . . . . :
Local network ID . . . . . . . . . . . . . . . . :  NETID
Local control point name . . . . . . . . . . . . :  AS400PU
Default local location . . . . . . . . . . . . . :  AS400LU
Default mode . . . . . . . . . . . . . . . . . . :  BLANK
APPN node type . . . . . . . . . . . . . . . . . :  *ENDNODE
Data compression . . . . . . . . . . . . . . . . :  *NONE
Intermediate data compression . . . . . . . . . :  *NONE
Maximum number of intermediate sessions . . . . :  200
Route addition resistance . . . . . . . . . . . :  128
Server network ID/control point name . . . . . . :  NETID    NETCP




More...
Press Enter to continue.

F3=Exit  F12=Cancel
```

Check that the values for **Local network ID** (1), **Local control point name** (2), and **Default local location** (3), correspond to the values on your worksheet.

IBM i *How to find the value of Resource name*

To find the value of resource name, type `WRKHDWRSC TYPE(*CMN)` and press enter.

The Work with Communication Resources panel is displayed. The value for **Resource name** is found as the token-ring Port. It is LIN041 in this example.

```
Work with Communication Resources
System:  AS400PU
Type options, press Enter.
2=Edit  4=Remove  5=Work with configuration description
7=Add configuration description ...

Configuration
Opt Resource     Description  Type Description
CC02              2636 Comm Processor
LIN04            2636 LAN Adapter
LIN041    TOKEN-RING  2636 Token-ring Port









Bottom
F3=Exit   F5=Refresh  F6=Print  F11=Display resource addresses/statuses
F12=Cancel  F23=More options
```

IBM i Establishing an LU 6.2 connection

This section describes how to establish an LU 6.2 connection.

IBM i *Local node configuration*

To configure the local node you need to create a line description and add a routing entry.

Creating a line description

1. If the line description has not already been created use the command CRTLINTRN.
2. Specify values for **Line description** (6) and **Resource name** (7).

```
Create Line Desc (token-ring) (CRTLINTRN)

Type choices, press Enter.

Line description . . . . . .  TOKENRINGL  Name
Resource name . . . . . . .  LIN041    Name, *NWID
NWI type . . . . . . . . . .  *FR      *FR, *ATM
Online at IPL . . . . . . .  *YES     *YES, *NO
Vary on wait . . . . . . . .  *NOWAIT    *NOWAIT, 15-180 (1 second)
Maximum controllers . . . . .  40      1-256
Attached NWI . . . . . . . .  *NONE     Name, *NONE



Bottom
F3=Exit  F4=Prompt  F5=Refresh  F10=Additional parameters  F12=Cancel
F13=How to use this display    F24=More keys
Parameter LIND required.                          +
```

Adding a routing entry

1. Type the command ADDRTGE and press enter.

```
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, * LI BL, *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
Parameter SBSD required.                          +
```

2. Specify your value for **Subsystem description** (5), and the values shown here for **Routing entry sequence number**, **Compare value** (8), **Starting position**, **Program to call**, and the **Library** containing the program to call.
3. Type the command STRSBS *subsystem description* (5) and press enter.

IBM i *Connection to partner node*

To connect to a partner node, you need to: create a controller description, create a device description, create CPI-C side information, add a communications entry for APPC, and add a configuration list entry.

This example is for a connection to a Windows system, but the steps are the same for other nodes.

Creating a controller description

1. At a command-line, type CRTCTLAPPC and press enter.

```
Create Ctl Desc (APPC) (CRTCTLAPPC)

Type choices, press Enter.

Controller description . . . . .  WINNTCP     Name
Link type . . . . . . . . . . .  *LAN     *FAX, *FR, *IDLC,
*LAN...
Online at IPL . . . . . . . . .  *NO      *YES, *NO



Bottom
F3=Exit  F4=Prompt  F5=Refresh  F10=Additional parameters  F12=Cancel
F13=How to use this display    F24=More keys
Parameter CTLD required.                          +
```

2. Specify a value for **Controller description** (12), set **Link type** to *LAN, and set **Online at IPL** to *NO.
3. Press enter twice, followed by F10.

```
Create Ctl Desc (APPC) (CRTCTLAPPC)

Type choices, press Enter.

Controller description . . . . . > WINNTCP     Name
Link type . . . . . . . . . . . > *LAN     *FAX, *FR, *IDLC, *LAN...
Online at IPL . . . . . . . . . > *NO      *YES, *NO
APPN-capable . . . . . . . . . .  *YES     *YES, *NO
Switched line list . . . . . . .  TOKENRINGL  Name
+ for more values
Maximum frame size . . . . . . .  *LINKTYPE   265-16393, 256, 265, 512...
Remote network identifier . . .  NETID     Name, *NETATR, *NONE, *ANY
Remote control point . . . . . .  WINNTCP     Name, *ANY
Exchange identifier . . . . . .         00000000-FFFFFFFF
Initial connection . . . . . . .  *DIAL     *DIAL, *ANS
Dial initiation . . . . . . . .  *LINKTYPE   *LINKTYPE, *IMMED, *DELAY
LAN remote adapter address . . .  10005AFC5D83 000000000001-FFFFFFFFFFFF
APPN CP session support . . . .  *YES     *YES, *NO
APPN node type . . . . . . . . .  *ENDNODE   *ENDNODE, *LENNODE...
APPN transmission group number   1       1-20, *CALC
More...
F3=Exit  F4=Prompt  F5=Refresh  F12=Cancel  F13=How to use this display
F24=More keys
```

4. Specify values for **Switched line list** (6), **Remote network identifier** (9), **Remote control point** (10), and **LAN remote adapter address** (16).
5. Press enter.

Creating a device description

1. Type the command CRTDEVAPPC and press enter.

```
Create Device Desc (APPC) (CRTDEVAPPC)

Type choices, press Enter.

Device description . . . . . .  WINNTLU     Name
Remote location . . . . . . .  WINNTLU     Name
Online at IPL . . . . . . . .  *YES     *YES, *NO
Local location . . . . . . . .  AS400LU    Name, *NETATR
Remote network identifier . . .  NETID     Name, *NETATR, *NONE
Attached controller . . . . . .  WINNTCP     Name
Mode . . . . . . . . . . . . .  *NETATR    Name, *NETATR
+ for more values
Message queue . . . . . . . .  QSYSOPR    Name, QSYSOPR
Library . . . . . . . . . . .   *LIBL    Name, *LIBL, *CURLIB
APPN-capable . . . . . . . . .  *YES     *YES, *NO
Single session:
Single session capable . . . .  *NO      *NO, *YES
Number of conversations . . .         1-512


Bottom
F3=Exit  F4=Prompt  F5=Refresh  F10=Additional parameters  F12=Cancel
F13=How to use this display    F24=More keys
Parameter DEVD required.                           +
```

2. Specify values for **Device description** (13), **Remote location** (11), **Local location** (3), **Remote network identifier** (9), and **Attached controller** (12).

Note: You can avoid having to create controller and device descriptions manually by taking advantage of the IBM i auto-configuration service. Consult the IBM i documentation for details.

Creating CPI-C side information

1. Type CRTCSI and press F10.

```
Create Comm Side Information (CRTCSI)

Type choices, press Enter.

Side information . . . . . .  NTCPIC    Name
Library . . . . . . . . . .   *CURLIB   Name, *CURLIB
Remote location . . . . . .  WINNTLU     Name
Transaction program . . . . .  MQSERIES

Text 'description' . . . . . .  *BLANK


Additional Parameters

Device . . . . . .  *LOC     Name, *LOC
Local location . . . . . .  AS400LU    Name, *LOC, *NETATR
Mode . . . . . .  #INTER    Name, *NETATR
Remote network identifier . . .  NETID     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
Parameter CSI required.
```

2. Specify values for **Side information** (14), **Remote location** (11), **Transaction program** (15), **Local location** (3), **Mode**, and **Remote network identifier** (9).
3. Press enter.

Adding a communications entry for APPC

1. At a command-line, type ADDCMNE and press enter.

```
Add Communications Entry (ADDCMNE)

Type choices, press Enter.

Subsystem description . . . . .  QCMN     Name
Library . . . . . .   *LIBL    Name, *LIBL, *CURLIB
Device . . . . . .  WINNTLU     Name, generic*, *ALL...
Remote location . . . . . .         Name
Job description . . . . . .  *USRPRF    Name, *USRPRF, *SBSD
Library . . . . . .         Name, *LIBL, *CURLIB
Default user profile . . . . . .  *NONE     Name, *NONE, *SYS
Mode . . . . . .  *ANY     Name, *ANY
Maximum active jobs . . . . . .  *NOMAX    0-1000, *NOMAX



Bottom
F3=Exit  F4=Prompt  F5=Refresh  F12=Cancel  F13=How to use this display
F24=More keys
Parameter SBSD required.
```

2. Specify values for **Subsystem description** (5) and **Device** (13), and press enter.

Adding a configuration list entry

1. Type ADDCFGLE *APPNRMT and press F4.

```
Add Configuration List Entries (ADDCFGLE)

Type choices, press Enter.

Configuration list type . . . . > *APPNRMT   *APPNLCL, *APPNRMT...
APPN remote location entry:
Remote location name . . . . .  WINNTLU     Name, generic*, *ANY
Remote network identifier . .  NETID     Name, *NETATR, *NONE
Local location name . . . . .  AS400LU    Name, *NETATR
Remote control point . . . . .  WINNTCP    Name, *NONE
Control point net ID . . . . .  NETID     Name, *NETATR, *NONE
Location password . . . . . .  *NONE
Secure location . . . . . . .  *NO      *YES, *NO
Single session . . . . . . . .  *NO      *YES, *NO
Locally controlled session . .  *NO      *YES, *NO
Pre-established session . . .  *NO      *YES, *NO
Entry 'description' . . . . .  *BLANK
Number of conversations . . .  10      1-512
+ for more values

Bottom
F3=Exit  F4=Prompt  F5=Refresh  F12=Cancel  F13=How to use this display
F24=More keys
```

2. Specify values for **Remote location name** (11), **Remote network identifier** (9), **Local location name** (3), **Remote control point** (10), and **Control point net ID** (9).
3. Press enter.

IBM i *What next?*

The LU 6.2 connection is now established. You are ready to complete the configuration.

Go to "IBM MQ for IBM i configuration" on page 31.

IBM i Establishing a TCP connection

If TCP is already configured there are no extra configuration tasks. If TCP/IP is not configured you need to: add a TCP/IP interface, add a TCP/IP loopback interface, and add a default route.

Adding a TCP/IP interface

1. At a command-line, type ADDTCPIFC and press enter.

```
Add TCP/IP Interface (ADDTCPIFC)

Type choices, press Enter.

Internet address . . . . . . .  19.22.11.55
Line description . . . . . . .  TOKENRINGL  Name, *LOOPBACK
Subnet mask . . . . . . . . . .  255.255.0.0
Type of service . . . . . . . .  *NORMAL    *MINDELAY, *MAXTHRPUT..
Maximum transmission unit . . .  *LIND     576-16388, *LIND
Autostart . . . . . . . . . . .  *YES     *YES, *NO
PVC logical channel identifier          001-FFF
+ for more values
X.25 idle circuit timeout . . .  60      1-600
X.25 maximum virtual circuits .  64      0-64
X.25 DDN interface . . . . . . .  *NO      *YES, *NO
TRLAN bit sequencing . . . . . .  *MSB     *MSB, *LSB




Bottom
F3=Exit  F4=Prompt  F5=Refresh  F12=Cancel  F13=How to use this display
F24=More keys
```

2. Specify the **IP address** and **Line description**, and a **Subnet mask** of the machine.
3. Press enter.

Adding a TCP/IP loopback interface

1. At a command-line, type ADDTCPIFC and press enter.

```
Add TCP Interface (ADDTCPIFC)

Type choices, press Enter.

Internet address . . . . . . . .  127.0.0.1
Line description . . . . . . . .  *LOOPBACK   Name, *LOOPBACK
Subnet mask . . . . . . . . . .  255.0.0.0
Type of service . . . . . . . .  *NORMAL    *MINDELAY, *MAXTHRPUT..
Maximum transmission unit . . .  *LIND     576-16388, *LIND
Autostart . . . . . . . . . . .  *YES     *YES, *NO
PVC logical channel identifier          001-FFF
+ for more values
X.25 idle circuit timeout . . .  60      1-600
X.25 maximum virtual circuits .  64      0-64
X.25 DDN interface . . . . . . .  *NO      *YES, *NO
TRLAN bit sequencing . . . . . .  *MSB     *MSB, *LSB



Bottom
F3=Exit  F4=Prompt  F5=Refresh  F12=Cancel  F13=How to use this display
F24=More keys
```

2. Specify the values for **IP address**, **Line description**, and **Subnet mask**.

Adding a default route

1. At a command-line, type ADDTCPRTE and press enter.

```
Add TCP Route (ADDTCPRTE)

Type choices, press Enter.

Route destination . . . . . . .  *DFTROUTE
Subnet mask . . . . . . . . . .  *NONE
Type of service . . . . . . . .  *NORMAL    *MINDELAY, *MAXTHRPUT.
Next hop . . . . . . . . . . .  19.2.3.4
Maximum transmission unit . . .  576      576-16388, *IFC



Bottom
F3=Exit  F4=Prompt  F5=Refresh  F12=Cancel  F13=How to use this display
F24=More keys
Command prompting ended when user pressed F12.
```

2. Enter values appropriate to your network and press enter to create a default route entry.

What next?

The TCP connection is now established. You are ready to complete the configuration. Go to "IBM MQ for IBM i configuration" on page 31.

IBM i IBM MQ for IBM i configuration

To configure IBM MQ for IBM i, use the WRKMQMQ command to display the configuration menu.

Start the TCP channel listener using the command STRMQMLSR.

Start any sender channel using the command STRMQMCHL CHLNAME(*channel_name*).

Use the WRKMQMQ command to display the IBM MQ configuration menu.

Note: AMQ* errors are placed in the log relating to the job that found the error. Use the WRKACTJOB command to display the list of jobs. Under the subsystem name QSYSWRK, locate the job and enter 5 against it to work with that job. IBM MQ logs are prefixed AMQ.

IBM i *Creating a queue manager*

Use the following steps to set up the basic configuration queue manager.

1. First you need to create a queue manager. Type CRTMQM and press enter.

```
                      Create Message Queue Manager (CRTMQM)

Type choices, press Enter.

Message Queue Manager name . . .

Text 'description' . . . . . . .   *BLANK

Trigger interval . . . . . . . .   999999999     0-999999999
Undelivered message queue  . . .   *NONE

Default transmission queue . . .   *NONE

Maximum handle limit . . . . . .   256           1-999999999
Maximum uncommitted messages . .   1000          1-10000
Default Queue manager  . . . . .   *NO           *YES, *NO




                                                                      Bottom
F3=Exit   F4=Prompt   F5=Refresh   F12=Cancel   F13=How to use this display
F24=More keys
```

2. In the **Message Queue Manager name** field, type AS400. In the **Undelivered message queue** field, type DEAD.LETTER.QUEUE.
3. Press enter.
4. Now start the queue manager by entering STRMQM MQMNAME(AS400).
5. Create the undelivered message queue using the following parameters. (For details and an example refer to "Defining a queue" on page 32.)

```
Local Queue
     Queue name :   DEAD.LETTER.QUEUE
     Queue type :   *LCL
```

IBM i *Defining a queue*

You can define a queue using the CRTMQMQ command.

Type CRTMQMQ on the command line.

```
Create MQM Queue (CRTMQMQ)

Type choices, press Enter.

Queue name . . . . . .

Queue type . . . . . .         *ALS, *LCL, *RMT









Bottom
F3=Exit  F4=Prompt  F5=Refresh  F12=Cancel  F13=How to use this display
F24=More keys
Parameter QNAME required.
```

Complete the two fields of this panel and press enter. Another panel is shown, with entry fields for the other parameters you have. Defaults can be taken for all other queue attributes.

IBM i *Defining a channel on IBM i*

On IBM i, you can define a channel using the CRTMQMCHL command.

Type CRTMQMCHL on the command line.

```
Create MQM Channel (CRTMQMCHL)

Type choices, press Enter.

Channel name . . . . . .
Channel type . . . . . .         *RCVR, *SDR, *SVR, *RQSTR









Bottom
F3=Exit  F4=Prompt  F5=Refresh  F12=Cancel  F13=How to use this display
F24=More keys
Parameter CHLNAME required.
```

Complete the two fields of this panel and press enter. Another panel is displayed on which you can specify the values for the other parameters given earlier. Defaults can be taken for all other channel attributes.

IBM i *Channel configuration for IBM i*

You need to configure your channels to implement the example configuration channels.

This section details the configuration to be performed on the IBM i queue manager to implement the channel described in “Example IBM MQ configuration for all platforms” on page 5.

Examples are given for connecting IBM MQ for IBM i and IBM MQ for Windows. To connect to IBM MQ on another platform, use the appropriate values from the table in place of those values for Windows

Note:

1. The words in **bold** are suggested values and reflect the names of IBM MQ objects used throughout these examples. You can change them in your product installation but, if you do, make sure that you use your own values when working through the examples in this section.
2. The IBM MQ channel ping command (PNGMQMCHL) runs interactively, whereas starting a channel causes a batch job to be submitted. If a channel ping completes successfully but the channel does not start, the network and IBM MQ definitions are probably correct, but that the IBM i environment for the batch job is not. For example, make sure that QSYS2 is included in the system portion of the library list and not just your personal library list.

For details and examples of how to create the objects listed refer to "Defining a queue" on page 32 and "Defining a channel on IBM i" on page 33.

Table 4. Configuration examples for IBM i

ID	Parameter Name	Reference	Example Used
Definition for local node			
A	Queue Manager Name		**AS400**
B	Local queue name		**AS400.LOCALQ**
Windows Windows ***Connection to IBM MQ for Windows*** The values in this section of the table must match the values used in "Channel configuration for Windows" on page 52, as indicated.			
C	Remote queue manager name	A	**WINNT**
D	Remote queue name		**WINNT.REMOTEQ**
E	Queue name at remote system	B	**WINNT.LOCALQ**
F	Transmission queue name		**WINNT**
G	Sender (SNA) channel name		**AS400.WINNT.SNA**
H	Sender (TCP/IP) channel name		**AS400.WINNT.TCP**
I	Receiver (SNA) channel name	G	**WINNT.AS400.SNA**
J	Receiver (TCP/IP) channel name	H	**WINNT.AS400.TCP**
AIX AIX ***Connection to IBM MQ for AIX*** The values in this section of the table must match the values used in "Channel configuration for AIX" on page 11, as indicated.			
C	Remote queue manager name	A	**AIX**
D	Remote queue name		**AIX.REMOTEQ**
E	Queue name at remote system	B	**AIX.LOCALQ**
F	Transmission queue name		**AIX**
G	Sender (SNA) channel name		**AS400.AIX.SNA**
H	Sender (TCP/IP) channel name		**AS400.AIX.TCP**
I	Receiver (SNA) channel name	G	**AIX.AS400.SNA**

Table 4. Configuration examples for IBM i (continued)			
ID	**Parameter Name**	**Reference**	**Example Used**
J	Receiver (TCP) channel name	H	**AIX.AS400.TCP**
HP-UX HP-UX ***Connection to IBM MQ for HP-UX*** The values in this section of the table must match the values used in "Channel configuration for HP-UX" on page 17, as indicated.			
C	Remote queue manager name	A	**HPUX**
D	Remote queue name		**HPUX.REMOTEQ**
E	Queue name at remote system	B	**HPUX.LOCALQ**
F	Transmission queue name		**HPUX**
G	Sender (SNA) channel name		**AS400.HPUX.SNA**
H	Sender (TCP) channel name		**AS400.HPUX.TCP**
I	Receiver (SNA) channel name	G	**HPUX.AS400.SNA**
J	Receiver (TCP) channel name	H	**HPUX.AS400.TCP**
Solaris Solaris ***Connection to IBM MQ for Solaris*** The values in this section of the table must match the values used in "Channel configuration for Solaris" on page 45, as indicated.			
C	Remote queue manager name	A	**SOLARIS**
D	Remote queue name		**SOLARIS.REMOTEQ**
E	Queue name at remote system	B	**SOLARIS.LOCALQ**
F	Transmission queue name		**SOLARIS**
G	Sender (SNA) channel name		**AS400.SOLARIS.SNA**
H	Sender (TCP/IP) channel name		**AS400.SOLARIS.TCP**
I	Receiver (SNA) channel name	G	**SOLARIS.AS400.SNA**
J	Receiver (TCP/IP) channel name	H	**SOLARIS.AS400.TCP**
Linux Linux ***Connection to IBM MQ for Linux*** The values in this section of the table must match the values used in "Channel configuration for Linux" on page 40, as indicated.			
C	Remote queue manager name	A	**LINUX**
D	Remote queue name		**LINUX.REMOTEQ**
E	Queue name at remote system	B	**LINUX.LOCALQ**
F	Transmission queue name		**LINUX**
G	Sender (SNA) channel name		**AS400.LINUX.SNA**
H	Sender (TCP/IP) channel name		**AS400.LINUX.TCP**
I	Receiver (SNA) channel name	G	**LINUX.AS400.SNA**
J	Receiver (TCP/IP) channel name	H	**LINUX.AS400.TCP**

Table 4. Configuration examples for IBM i (continued)			
ID	**Parameter Name**	**Reference**	**Example Used**
z/OS z/OS ***Connection to IBM MQ for z/OS*** The values in this section of the table must match the values used in "Channel configuration for z/OS" on page 57, as indicated.			
C	Remote queue manager name	A	**MVS**
D	Remote queue name		**MVS.REMOTEQ**
E	Queue name at remote system	B	**MVS.LOCALQ**
F	Transmission queue name		**MVS**
G	Sender (SNA) channel name		**AS400.MVS.SNA**
H	Sender (TCP) channel name		**AS400.MVS.TCP**
I	Receiver (SNA) channel name	G	**MVS.AS400.SNA**
J	Receiver (TCP) channel name	H	**MVS.AS400.TCP**
Connection to MQSeries for VSE/ESA The values in this section of the table must match the values used in your VSE/ESA system.			
C	Remote queue manager name	A	**VSE**
D	Remote queue name		**VSE.REMOTEQ**
E	Queue name at remote system	B	**VSE.LOCALQ**
F	Transmission queue name		**VSE**
G	Sender channel name		**AS400.VSE.SNA**
I	Receiver channel name	G	**VSE.AS400.SNA**

IBM i *Sender-channel definitions for IBM i*
Example sender-channel definitions for SNA and TCP.

Using SNA

```
     Local Queue
              Queue name :   WINNT                            F
              Queue type :   *LCL
                   Usage :   *TMQ

     Remote Queue
              Queue name :   WINNT.REMOTEQ                    D
              Queue type :   *RMT
             Remote queue :  WINNT.LOCALQ                     E
   Remote Queue Manager :   WINNT                             C
     Transmission queue :   WINNT                             F

     Sender Channel
             Channel Name :  AS400.WINNT.SNA                  G
             Channel Type :  *SDR
          Transport type :   *LU62
         Connection name :   WINNTCPIC                        14
     Transmission queue :   WINNT                             F
```

Using TCP

```
Local Queue
         Queue name :   WINNT                            F
         Queue type :   *LCL
              Usage :   *TMQ

Remote Queue
         Queue name :   WINNT.REMOTEQ                    D
         Queue type :   *RMT
        Remote queue :  WINNT.LOCALQ                     E
Remote Queue Manager :  WINNT                            C
  Transmission queue :  WINNT                            F

Sender Channel
        Channel Name :  AS400.WINNT.TCP                  H
        Channel Type :  *SDR
      Transport type :  *TCP
     Connection name :  WINNT.tcpip.hostname
  Transmission queue :  WINNT                            F
```

IBM i *Receiver-channel definitions for IBM i*

Example receiver-channel definitions for SNA and TCP

Using SNA

```
Local Queue
         Queue name :   AS400.LOCALQ                   B
         Queue type :   *LCL

Receiver Channel
        Channel Name :  WINNT.AS400.SNA                I
        Channel Type :  *RCVR
      Transport type :  *LU62
```

Using TCP

```
Local Queue
         Queue name :   AS400.LOCALQ                   B
         Queue type :   *LCL

Receiver Channel
        Channel Name :  WINNT.AS400.TCP                J
        Channel Type :  *RCVR
      Transport type :  *TCP
```

Linux Example MQ configuration for Linux

This section gives an example of how to set up communication links from IBM MQ for Linux to IBM MQ products.

The examples given are on the following platforms:

- Windows
- AIX
- Compaq Tru64 UNIX
- HP-UX
- Solaris
- IBM i
- z/OS
- VSE/ESA

See “Example IBM MQ configuration for all platforms” on page 5 for background information about this section and how to use it.

Linux Establishing an LU 6.2 connection

Use this worksheet to record the values you use for your configuration.

Note: The information in this section applies only to IBM MQ for Linux (x86 platform). It does not apply to IBM MQ for Linux (x86-64 platform), IBM MQ for Linux (zSeries s390x platform), or IBM MQ for Linux (Power platform).

For the latest information about configuring SNA over TCP/IP, refer to the the Administration Guide for your version of Linux from the following documentation: Communications Server for Linux library.

Linux Establishing a TCP connection on Linux

Some Linux distributions now use the extended inet daemon (XINETD) instead of the inet daemon (INETD). The following instructions tell you how to establish a TCP connection using either the inet daemon or the extended inet daemon.

Using the inet daemon (INETD)

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

To establish a TCP connection, follow these steps.

1. Edit the file /etc/services. If you do not have the following line in the file, add it as shown:

   ```
   MQSeries    1414/tcp   # MQSeries channel listener
   ```

   Note: To edit this file, you must be logged in as a superuser or root.
2. 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 ]
   ```
3. Find the process ID of the inetd with the command:

   ```
   ps -ef | grep inetd
   ```
4. Run the command:

   ```
   kill -1 inetd processid
   ```

If you have more than one queue manager on your system, and therefore require more than one service, you must add a line for each additional queue manager to both /etc/services and inetd.conf.

For example:

```
MQSeries1   1414/tcp
MQSeries2   1822/tcp
```

```
MQSeries1 stream tcp nowait mqm MQ_INSTALLATION_PATH/bin/amqcrsta amqcrsta -m QM1
MQSeries2 stream tcp nowait mqm MQ_INSTALLATION_PATH/bin/amqcrsta amqcrsta -m QM2
```

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.

The inetd process on Linux can limit the rate of inbound connections on a TCP port. The default is 40 connections in a 60 second interval. If you need a higher rate, specify a new limit on the number of inbound connections in a 60 second interval by appending a period (.) followed by the new limit to the nowait parameter of the appropriate service in inetd.conf. For example, for a limit of 500 connections in a 60 second interval use:

```
MQSeries stream tcp nowait.500 mqm / MQ_INSTALLATION_PATH/bin/amqcrsta amqcrsta -m QM1
```

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

Using the extended inet daemon (XINETD)

The following instructions describe how the extended inet daemon is implemented on Red Hat Linux. If you are using a different Linux distribution, you might have to adapt these instructions.

To establish a TCP connection, follow these steps.

1. Edit the file /etc/services. If you do not have the following line in the file, add it as shown:

   ```
   MQSeries    1414/tcp   # MQSeries channel listener
   ```

   Note: To edit this file, you must be logged in as a superuser or root.

2. Create a file called IBM MQ in the XINETD configuration directory, /etc/xinetd.d. Add the following stanza to the file:

   ```
   # IBM MQ service for XINETD
   service MQSeries
   {
     disable         = no
     flags           = REUSE
     socket_type     = stream
     wait            = no
     user            = mqm
     server          = MQ_INSTALLATION_PATH/bin/amqcrsta
     server_args     = -m queue.manager.name
     log_on_failure += USERID
   }
   ```

3. Restart the extended inet daemon by issuing the following command:

   ```
   /etc/rc.d/init.d/xinetd restart
   ```

If you have more than one queue manager on your system, and therefore require more than one service, you must add a line to /etc/services for each additional queue manager. You can create a file in the /etc/xinetd.d directory for each service, or you can add additional stanzas to the IBM MQ file you created previously.

The xinetd process on Linux can limit the rate of inbound connections on a TCP port. The default is 50 connections in a 10 second interval. If you need a higher rate, specify a new limit on the rate of inbound connections by specifying the 'cps' attribute in the xinetd configuration file. For example, for a limit of 500 connections in a 60 second interval use:

```
cps = 500 60
```

What next?

The TCP/IP connection is now established. You are ready to complete the configuration. Go to “IBM MQ for Linux configuration” on page 40.

Linux IBM MQ for Linux configuration

Before beginning the installation process ensure that you have first created the mqm user ID and the mqm group, and set the password.

Start any channel using the command:

```
runmqchl -c channel.name
```

Note:

1. Sample programs are installed in *MQ_INSTALLATION_PATH*/samp, where *MQ_INSTALLATION_PATH* represents the high-level directory in which IBM MQ is installed.
2. Error logs are stored in /var/mqm/qmgrs/ *qmgrname* /errors.
3. When you are using the command interpreter **runmqsc** to enter administration commands, a + at the end of a line indicates that the next line is a continuation. Ensure that there is a space between the last parameter and the continuation character.

Basic configuration

1. Create the queue manager from the UNIX prompt using the command:

   ```
   crtmqm -u dlqname -q linux
   ```

   where:

   linux
   Is the name of the queue manager

   -q
   Indicates that this is to become the default queue manager

   -u *dlqname*
   Specifies the name of the dead letter queue

   This command creates a queue manager and a set of default objects.
2. Start the queue manager from the UNIX prompt using the command:

   ```
   strmqm linux
   ```

   where *linux* is the name given to the queue manager when it was created.

Linux *Channel configuration for Linux*

The following section details the configuration to be performed on the Linux queue manager to implement the channel described in "Example IBM MQ configuration for all platforms" on page 5.

The MQSC command to create each object is shown. Either start **runmqsc** from a UNIX prompt and enter each command in turn, or build the commands into a command file.

Examples are given for connecting IBM MQ for Linux and IBM MQ for HP-UX. To connect to IBM MQ on another platform use the appropriate set of values from the table in place of those for HP-UX.

Note: The words in **bold** are suggested values and reflect the names of IBM MQ objects used throughout these examples. You can change them in your product installation but, if you do, make sure that you use your own values when working through the examples in this section

Table 5. Configuration examples for IBM MQ for Linux			
ID	**Parameter Name**	**Reference**	**Example Used**
Definition for local node			
A	Queue Manager Name		**LINUX**
B	Local queue name		**LINUX.LOCALQ**
Windows Windows ***Connection to IBM MQ for Windows*** The values in this section of the table must match those used in "Channel configuration for Windows" on page 52, as indicated.			
C	Remote queue manager name	A	**WINNT**
D	Remote queue name		**WINNT.REMOTEQ**
E	Queue name at remote system	B	**WINNT.LOCALQ**
F	Transmission queue name		**WINNT**
G	Sender (SNA) channel name		**LINUX.WINNT.SNA**
H	Sender (TCP/IP) channel name		**LINUX.WINNT.TCP**
I	Receiver (SNA) channel name	G	**WINNT.LINUX.SNA**
J	Receiver (TCP) channel name	H	**WINNT.LINUX.TCP**
AIX AIX ***Connection to IBM MQ for AIX*** The values in this section of the table must match those used in "Channel configuration for AIX" on page 11, as indicated.			
C	Remote queue manager name	A	**AIX**
D	Remote queue name		**AIX.REMOTEQ**
E	Queue name at remote system	B	**AIX.LOCALQ**
F	Transmission queue name		**AIX**
G	Sender (SNA) channel name		**LINUX.AIX.SNA**
H	Sender (TCP) channel name		**LINUX.AIX.TCP**
I	Receiver (SNA) channel name	G	**AIX.LINUX.SNA**
J	Receiver (TCP) channel name	H	**AIX.LINUX.TCP**
HP-UX HP-UX ***Connection to IBM MQ for HP-UX*** The values in this section of the table must match those used in Table 2 on page 17, as indicated.			
C	Remote queue manager name	A	**HPUX**
D	Remote queue name		**HPUX.REMOTEQ**
E	Queue name at remote system	B	**HPUX.LOCALQ**
F	Transmission queue name		**HPUX**
G	Sender (SNA) channel name		**LINUX.HPUX.SNA**
H	Sender (TCP) channel name		**LINUX.HPUX.TCP**

Table 5. Configuration examples for IBM MQ for Linux (continued)			
ID	**Parameter Name**	**Reference**	**Example Used**
I	Receiver (SNA) channel name	G	**HPUX.LINUX.SNA**
J	Receiver (TCP/IP) channel name	H	**HPUX.LINUX.TCP**
Solaris Solaris ***Connection to IBM MQ for Solaris*** The values in this section of the table must match those used in Table 6 on page 46, as indicated.			
C	Remote queue manager name	A	**SOLARIS**
D	Remote queue name		**SOLARIS.REMOTEQ**
E	Queue name at remote system	B	**SOLARIS.LOCALQ**
F	Transmission queue name		**GIS**
G	Sender (SNA) channel name		**LINUX.SOLARIS.SNA**
H	Sender (TCP/IP) channel name		**LINUX.SOLARIS.TCP**
I	Receiver (SNA) channel name	G	**SOLARIS.LINUX.SNA**
J	Receiver (TCP/IP) channel name	H	**SOLARIS.LINUX.TCP**
IBM i IBM i ***Connection to IBM MQ for IBM i*** The values in this section of the table must match those used in Table 4 on page 34, as indicated.			
C	Remote queue manager name	A	**AS400**
D	Remote queue name		**AS400.REMOTEQ**
E	Queue name at remote system	B	**AS400.LOCALQ**
F	Transmission queue name		**AS400**
G	Sender (SNA) channel name		**LINUX.AS400.SNA**
H	Sender (TCP) channel name		**LINUX.AS400.TCP**
I	Receiver (SNA) channel name	G	**AS400.LINUX.SNA**
J	Receiver (TCP) channel name	H	**AS400.LINUX.TCP**
z/OS z/OS ***Connection to IBM MQ for z/OS*** The values in this section of the table must match those used in Table 8 on page 58, as indicated.			
C	Remote queue manager name	A	**MVS**
D	Remote queue name		**MVS.REMOTEQ**
E	Queue name at remote system	B	**MVS.LOCALQ**
F	Transmission queue name		**MVS**
G	Sender (SNA) channel name		**LINUX.MVS.SNA**
H	Sender (TCP) channel name		**LINUX.MVS.TCP**
I	Receiver (SNA) channel name	G	**MVS.LINUX.SNA**

Linux *IBM MQ for Linux (x86 platform) sender-channel definitions using SNA*

Example coding.

```
def ql (HPUX) +                                  F
    usage(xmitq) +
    replace

def qr (HPUX.REMOTEQ) +                          D
    rname(HPUX.LOCALQ) +                         E
    rqmname(HPUX) +                              C
    xmitq(HPUX) +                                F
    replace

def chl (LINUX.HPUX.SNA) chltype(sdr) +          G
    trptype(lu62) +
    conname('HPUXCPIC') +                        14
    xmitq(HPUX) +                                F
    replace
```

Linux *IBM MQ for Linux (x86 platform) receiver-channel definitions using SNA*

Example coding.

```
def ql (LINUX.LOCALQ) replace                    B

def chl (HPUX.LINUX.SNA) chltype(rcvr) +         I
    trptype(lu62) +
    replace
```

Linux *IBM MQ for Linux sender-channel definitions using TCP*

Example coding.

```
def ql (HPUX) +                                  F
    usage(xmitq) +
    replace

def qr (HPUX.REMOTEQ) +                          D
    rname(HPUX.LOCALQ) +                         E
    rqmname(HPUX) +                              C
    xmitq(HPUX) +                                F
    replace

def chl (LINUX.HPUX.TCP) chltype(sdr) +          H
    trptype(tcp) +
    conname(remote_tcpip_hostname) +
    xmitq(HPUX) +                                F
    replace
```

Linux *IBM MQ for Linux receiver-channel definitions using TCP/IP*

Example coding.

```
def ql (LINUX.LOCALQ) replace                    B

def chl (HPUX.LINUX.TCP) chltype(rcvr) +         J
    trptype(tcp) +
    replace
```

Solaris Example MQ configuration for Solaris

This section gives an example of how to set up communication links from IBM MQ for Solaris to IBM MQ products.

Examples are given on the following platforms:

- Windows

- AIX
- HP Tru64 UNIX
- HP-UX
- Linux
- IBM i
- z/OS
- VSE/ESA

See "Example IBM MQ configuration for all platforms" on page 5 for background information about this section and how to use it.

Solaris Establishing an LU 6.2 connection using SNAP-IX

Parameters for configuring an LU 6.2 connection using SNAP-IX.

For the latest information about configuring SNA over TCP/IP, refer to the following online IBM documentation: Communications Server, the following online MetaSwitch documentation: SNAP-IX Administration Guide, and the following online Sun documentation: Configuring Intersystem Communications (ISC).

Solaris Establishing a TCP connection

Information about configuring a TCP connection and next steps.

To establish a TCP connection, follow these steps.

1. Edit the file /etc/services.

   Note: To edit the /etc/services file, you must be logged in as a superuser or root. If you do not have the following line in that file, add it as shown:

   ```
   MQSeries       1414/tcp      # MQSeries channel listener
   ```

2. 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]
   ```

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

3. Find the process ID of the inetd with the command:

   ```
   ps -ef | grep inetd
   ```

4. Run the appropriate command, as follows:

   - For Solaris 9:

     ```
     kill -1 inetd processid
     ```

   - For Solaris 10 or later:

     ```
     inetconv
     ```

What next?

The TCP/IP connection is now established. You are ready to complete the configuration. Go to "IBM MQ for Solaris configuration" on page 45.

Solaris IBM MQ for Solaris configuration

Describes channels to be defined to complete the configuration.

Before beginning the installation process ensure that you have first created the *mqm* user and group, and set the password.

Start any channel using the command:

```
runmqchl -c channel.name
```

Note:

1. Sample programs are installed in *MQ_INSTALLATION_PATH*/samp.

   MQ_INSTALLATION_PATH represents the high-level directory in which IBM MQ is installed.
2. Error logs are stored in /var/mqm/qmgrs/ *qmgrname* /errors.
3. When you are using the command interpreter **runmqsc** to enter administration commands, a + at the end of a line indicates that the next line is a continuation. Ensure that there is a space between the last parameter and the continuation character.
4. For an SNA or LU6.2 channel, if you experience an error when you try to load the communications library, probably file liblu62.so cannot be found. A likely solution to this problem is to add its location, which is probably /opt/SUNWlu62, to LD_LIBRARY_PATH.

Basic configuration

1. Create the queue manager from the UNIX prompt using the command:

   ```
   crtmqm -u dlqname -q solaris
   ```

   where:

   solaris
   Is the name of the queue manager

   -q
   Indicates that this is to become the default queue manager

   -u *dlqname*
   Specifies the name of the undeliverable message queue

   This command creates a queue manager and a set of default objects.
2. Start the queue manager from the UNIX prompt using the command:

   ```
   strmqm solaris
   ```

   where *solaris* is the name given to the queue manager when it was created.

Solaris *Channel configuration for Solaris*

The following section details the configuration to be performed on the Solaris queue manager to implement a channel.

The configuration described is to implement the channel described in Figure 1 on page 5.

The MQSC command to create each object is shown. Either start **runmqsc** from a UNIX prompt and enter each command in turn, or build the commands into a command file.

Examples are given for connecting IBM MQ for Solaris and IBM MQ for Windows. To connect to IBM MQ on another platform use the appropriate set of values from the table in place of those for Windows.

Note: The words in **bold** are user-specified and reflect the names of IBM MQ objects used throughout these examples. If you change the names used here, ensure that you also change the other references made to these objects throughout this section. All others are keywords and should be entered as shown.

Table 6. Configuration worksheet for IBM MQ for Solaris

ID	Parameter Name	Reference	Example Used	User Value
Definition for local node				
A	Queue Manager Name		**SOLARIS**	
B	Local queue name		**SOLARIS.LOCALQ**	
Connection to IBM MQ for Windows The values in this section of the table must match those used in Table 7 on page 52, as indicated.				
C	Remote queue manager name	A	**WINNT**	
D	Remote queue name		**WINNT.REMOTEQ**	
E	Queue name at remote system	B	**WINNT.LOCALQ**	
F	Transmission queue name		**WINNT**	
G	Sender (SNA) channel name		**SOLARIS.WINNT.SNA**	
H	Sender (TCP/IP) channel name		**SOLARIS.WINNT.TCP**	
I	Receiver (SNA) channel name	G	**WINNT.SOLARIS.SNA**	
J	Receiver (TCP) channel name	H	**WINNT.SOLARIS.TCP**	
Connection to IBM MQ for AIX The values in this section of the table must match those used in Table 1 on page 11, as indicated.				
C	Remote queue manager name	A	**AIX**	
D	Remote queue name		**AIX.REMOTEQ**	
E	Queue name at remote system	B	**AIX.LOCALQ**	
F	Transmission queue name		**AIX**	
G	Sender (SNA) channel name		**SOLARIS.AIX.SNA**	
H	Sender (TCP) channel name		**SOLARIS.AIX.TCP**	
I	Receiver (SNA) channel name	G	**AIX.SOLARIS.SNA**	
J	Receiver (TCP) channel name	H	**AIX.SOLARIS.TCP**	
Connection to MQSeries for Compaq Tru64 Unix The values in this section of the table must match those used in your Compaq Tru64 UNIX system.				
C	Remote queue manager name	A	**DECUX**	
D	Remote queue name		**DECUX.REMOTEQ**	
E	Queue name at remote system	B	**DECUX.LOCALQ**	
F	Transmission queue name		**DECUX**	
H	Sender (TCP) channel name		**DECUX.SOLARIS.TCP**	

Table 6. Configuration worksheet for IBM MQ for Solaris (continued)				
ID	**Parameter Name**	**Reference**	**Example Used**	**User Value**
J	Receiver (TCP) channel name	H	**SOLARIS.DECUX.TCP**	
Connection to IBM MQ for HP-UX The values in this section of the table must match those used in Table 2 on page 17, as indicated.				
C	Remote queue manager name	A	**HPUX**	
D	Remote queue name		**HPUX.REMOTEQ**	
E	Queue name at remote system	B	**HPUX.LOCALQ**	
F	Transmission queue name		**HPUX**	
G	Sender (SNA) channel name		**SOLARIS.HPUX.SNA**	
H	Sender (TCP) channel name		**SOLARIS.HPUX.TCP**	
I	Receiver (SNA) channel name	G	**HPUX.SOLARIS.SNA**	
J	Receiver (TCP/IP) channel name	H	**HPUX.SOLARIS.TCP**	
Connection to IBM MQ for Linux The values in this section of the table must match those used in Table 5 on page 41, as indicated.				
C	Remote queue manager name	A	**LINUX**	
D	Remote queue name		**LINUX.REMOTEQ**	
E	Queue name at remote system	B	**LINUX.LOCALQ**	
F	Transmission queue name		**LINUX**	
G	Sender (SNA) channel name		**SOLARIS.LINUX.SNA**	
H	Sender (TCP/IP) channel name		**SOLARIS.LINUX.TCP**	
I	Receiver (SNA) channel name	G	**LINUX.SOLARIS.SNA**	
J	Receiver (TCP/IP) channel name	H	**LINUX.SOLARIS.TCP**	
IBM i ***Connection to IBM MQ for IBM i*** IBM i The values in this section of the table must match those used in Table 4 on page 34, as indicated.				
C	Remote queue manager name	A	**AS400**	
D	Remote queue name		**AS400.REMOTEQ**	
E	Queue name at remote system	B	**AS400.LOCALQ**	
F	Transmission queue name		**AS400**	
G	Sender (SNA) channel name		**SOLARIS.AS400.SNA**	
H	Sender (TCP) channel name		**SOLARIS.AS400.TCP**	
I	Receiver (SNA) channel name	G	**AS400.SOLARIS.SNA**	
IBM i IBM i J	Receiver (TCP) channel name	H	**AS400.SOLARIS.TCP**	

Table 6. Configuration worksheet for IBM MQ for Solaris (continued)				
ID	**Parameter Name**	**Reference**	**Example Used**	**User Value**
z/OS ***Connection to IBM MQ for z/OS*** z/OS The values in this section of the table must match those used in Table 8 on page 58, as indicated.				
C	Remote queue manager name	A	**MVS**	
D	Remote queue name		**MVS.REMOTEQ**	
E	Queue name at remote system	B	**MVS.LOCALQ**	
F	Transmission queue name		**MVS**	
G	Sender (SNA) channel name		**SOLARIS.MVS.SNA**	
H	Sender (TCP) channel name		**SOLARIS.MVS.TCP**	
I	Receiver (SNA) channel name	G	**MVS.SOLARIS.SNA**	
z/OS z/OS J	Receiver (TCP) channel name	H	**MVS.SOLARIS.TCP**	
Connection to MQSeries for VSE/ESA The values in this section of the table must match those used in your VSE/ESA system.				
C	Remote queue manager name	A	**VSE**	
D	Remote queue name		**VSE.REMOTEQ**	
E	Queue name at remote system	B	**VSE.LOCALQ**	
F	Transmission queue name		**VSE**	
G	Sender channel name		**SOLARIS.VSE.SNA**	
I	Receiver channel name	G	**VSE.SOLARIS.SNA**	

Solaris *IBM MQ for Solaris sender-channel definitions using SNAP-IX SNA*
Example coding.

```
def ql (WINNT) +                                    F
    usage(xmitq) +
    replace

def qr (WINNT.REMOTEQ) +                            D
    rname(WINNT.LOCALQ) +                           E
    rqmname(WINNT) +                                C
    xmitq(WINNT) +                                  F
    replace

def chl (SOLARIS.WINNT.SNA) chltype(sdr) +          G
    trptype(lu62) +
    conname('NTCPIC') +            14
    xmitq(WINNT) +                                  F
    replace
```

Solaris *IBM MQ for Solaris receiver-channel definitions using SNA*

Example coding.

```
def ql (SOLARIS.LOCALQ) replace                    B

def chl (WINNT.SOLARIS.SNA) chltype(rcvr) +        I
    trptype(lu62) +
    replace
```

Solaris *IBM MQ for Solaris sender-channel definitions using TCP*

Example coding.

```
def ql (WINNT) +                                     F
    usage(xmitq) +
    replace

def qr (WINNT.REMOTEQ) +                             D
    rname(WINNT.LOCALQ) +                            E
    rqmname(WINNT) +                                 C
    xmitq(WINNT) +                                   F
    replace

def chl (SOLARIS.WINNT.TCP) chltype(sdr) +           H
    trptype(tcp) +
    conname(remote_tcpip_hostname) +
    xmitq(WINNT) +                                   F
    replace
```

Solaris *IBM MQ for Solaris receiver-channel definitions using TCP/IP*

Example coding.

```
def ql (SOLARIS.LOCALQ) replace                    B

def chl (WINNT.SOLARIS.TCP) chltype(rcvr) +        J
    trptype(tcp) +
    replace
```

Windows Example IBM MQ configuration for Windows

This section gives an example of how to set up communication links from IBM MQ for Windows to IBM MQ products on other platforms.

Setup of communication links is shown on the following platforms:

- AIX
- HP Tru64 UNIX
- HP-UX
- Solaris
- Linux
- IBM i
- z/OS
- VSE/ESA

When the connection is established, you must define some channels to complete the configuration. Example programs and commands for configuration are described in "IBM MQ for Windows configuration" on page 51.

See "Example IBM MQ configuration for all platforms" on page 5 for background information about this section and how to use it.

Windows Establishing an LU 6.2 connection

Reference to information about configuring AnyNet® SNA over TCP/IP.

For the latest information about configuring AnyNet SNA over TCP/IP, see the following online IBM documentation: AnyNet SNA over TCP/IP, SNA Node Operations, and Communications Server for Windows

Windows Establishing a TCP connection

The TCP stack that is shipped with Windows systems does not include an *inet* daemon or equivalent.

The IBM MQ command used to start the IBM MQ for TCP listener is:

```
runmqlsr -t tcp
```

The listener must be started explicitly before any channels are started. It enables receiving channels to start automatically in response to a request from an inbound sending channel.

What next?

When the TCP/IP connection is established, you are ready to complete the configuration. Go to "IBM MQ for Windows configuration" on page 51.

Windows Establishing a NetBIOS connection

A NetBIOS connection is initiated from a queue manager that uses the ConnectionName parameter on its channel definition to connect to a target listener.

To set up a NetBIOS connection, follow these steps:

1. At each end of the channel specify the local NetBIOS name to be used by the IBM MQ channel processes in the queue manager configuration file qm.ini. For example, the NETBIOS stanza in Windows at the sending end might look like the following:

   ```
   NETBIOS:
    LocalName=WNTNETB1
   ```

   and at the receiving end:

   ```
   NETBIOS:
    LocalName=WNTNETB2
   ```

   Each IBM MQ process must use a different local NetBIOS name. Do not use your system name as the NetBIOS name because Windows already uses it.
2. At each end of the channel, verify the LAN adapter number being used on your system. The IBM MQ for Windows default for logical adapter number 0 is NetBIOS running over an Internet Protocol network. To use native NetBIOS you must select logical adapter number 1. See Establishing the LAN adapter number.

   Specify the correct LAN adapter number in the NETBIOS stanza of the Windows registry. For example:

   ```
   NETBIOS:
    AdapterNum=1
   ```
3. So that sender channel initiation works, specify the local NetBIOS name by the MQNAME environment variable:

   ```
    SET MQNAME=WNTNETB1I
   ```

This name must be unique.

4. At the sending end, define a channel specifying the NetBIOS name being used at the other end of the channel. For example:

```
DEFINE CHANNEL (WINNT.OS2.NET) CHLTYPE(SDR) +
       TRPTYPE(NETBIOS) +
       CONNAME(WNTNETB2) +
       XMITQ(OS2) +
       MCATYPE(THREAD) +
       REPLACE
```

   You must specify the option `MCATYPE(THREAD)` because, on Windows, sender channels must be run as threads.

5. At the receiving end, define the corresponding receiver channel. For example:

```
DEFINE CHANNEL (WINNT.OS2.NET) CHLTYPE(RCVR) +
       TRPTYPE(NETBIOS) +
       REPLACE
```

6. Start the channel initiator because each new channel is started as a thread rather than as a new process.

```
runmqchi
```

7. At the receiving end, start the IBM MQ listener:

```
runmqlsr -t netbios
```

   Optionally you can specify values for the queue manager name, NetBIOS local name, number of sessions, number of names, and number of commands. See Defining a NetBIOS connection on Windows for more information about setting up NetBIOS connections.

Windows IBM MQ for Windows configuration

Example programs and commands for configuration.

Note:

1. You can use the sample program, AMQSBCG, to show the contents and headers of all the messages in a queue. For example:

```
AMQSBCG q_name qmgr_name
```

   shows the contents of the queue *q_name* defined in queue manager *qmgr_name*.

   Alternatively, you can use the message browser in the IBM MQ Explorer.

2. You can start any channel from the command prompt using the command

```
runmqchl -c channel.name
```

3. Error logs can be found in the directories *MQ_INSTALLATION_PATH*\qmgrs\ *qmgrname* \errors and *MQ_INSTALLATION_PATH*\qmgrs\@system\errors. In both cases, the most recent messages are at the end of amqerr01.log.

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

4. When you are using the command interpreter **runmqsc** to enter administration commands, a + at the end of a line indicates that the next line is a continuation. Ensure that there is a space between the last parameter and the continuation character.

Windows Default configuration

You can create a default configuration by using the IBM MQ Postcard application to guide you through the process.

For information about using the Postcard application, see *Verify the installation using the Postcard application* for the platform that your enterprise uses.

Windows Basic configuration

You can create and start a queue manager from the IBM MQ Explorer or from the command prompt.

.If you choose the command prompt:

1. Create the queue manager using the command:

   ```
   crtmqm -u dlqname -q winnt
   ```

   where:

   winnt
   Is the name of the queue manager

   -q
   Indicates that this is to become the default queue manager

   -u *dlqname*
   Specifies the name of the undeliverable message queue

   This command creates a queue manager and a set of default objects.

2. Start the queue manager using the command:

   ```
   strmqm winnt
   ```

   where *winnt* is the name given to the queue manager when it was created.

Windows Channel configuration for Windows

Example configuration to be performed on the Windows queue manager to implement a given channel.

The following sections detail the configuration to be performed on the Windows queue manager to implement the channel described in "Example IBM MQ configuration for all platforms" on page 5.

In each case the MQSC command is shown. Either start **runmqsc** from a command prompt and enter each command in turn, or build the commands into a command file.

Examples are given for connecting IBM MQ for Windows and IBM MQ for AIX. To connect to IBM MQ on another platform use the appropriate set of values from the table in place of those for Windows.

Note: The words in **bold** are suggested values and reflect the names of IBM MQ objects used throughout these examples. You can change them in your product installation but, if you do, make sure that you use your own values when working through the examples in this section.

Table 7. Configuration examples for IBM MQ for Windows

	Parameter Name	Reference	Example Used
Definition for local node			
A	Queue Manager Name		**WINNT**
B	Local queue name		**WINNT.LOCALQ**

Table 7. Configuration examples for IBM MQ for Windows (continued)			
	Parameter Name	**Reference**	**Example Used**
AIX AIX ***Connection to IBM MQ for AIX*** The values in this section of the table must match those used in "Channel configuration for AIX" on page 11, as indicated.			
C	Remote queue manager name	A	**AIX**
D	Remote queue name		**AIX.REMOTEQ**
E	Queue name at remote system	B	**AIX.LOCALQ**
F	Transmission queue name		**AIX**
G	Sender (SNA) channel name		**WINNT.AIX.SNA**
H	Sender (TCP) channel name		**WINNT.AIX.TCP**
I	Receiver (SNA) channel name	G	**AIX.WINNT.SNA**
J	Receiver (TCP) channel name	H	**AIX.WINNT.TCP**
HP-UX HP-UX ***Connection to IBM MQ for HP-UX*** The values in this section of the table must match those used in "Channel configuration for HP-UX" on page 17, as indicated.			
C	Remote queue manager name	A	**HPUX**
D	Remote queue name		**HPUX.REMOTEQ**
E	Queue name at remote system	B	**HPUX.LOCALQ**
F	Transmission queue name		**HPUX**
G	Sender (SNA) channel name		**WINNT.HPUX.SNA**
H	Sender (TCP) channel name		**WINNT.HPUX.TCP**
I	Receiver (SNA) channel name	G	**HPUX.WINNT.SNA**
J	Receiver (TCP/IP) channel name	H	**HPUX.WINNT.TCP**
Solaris Solaris ***Connection to IBM MQ for Solaris*** The values in this section of the table must match those used in "Channel configuration for Solaris" on page 45, as indicated.			
C	Remote queue manager name	A	**SOLARIS**
D	Remote queue name		**SOLARIS.REMOTEQ**
E	Queue name at remote system	B	**SOLARIS.LOCALQ**
F	Transmission queue name		**SOLARIS**
G	Sender (SNA) channel name		**WINNT.SOLARIS.SNA**
H	Sender (TCP) channel name		**WINNT.SOLARIS.TCP**
I	Receiver (SNA) channel name	G	**SOLARIS.WINNT.SNA**
J	Receiver (TCP) channel name	H	**SOLARIS.WINNT.TCP**

Table 7. Configuration examples for IBM MQ for Windows (continued)			
	Parameter Name	**Reference**	**Example Used**
Linux Linux ***Connection to IBM MQ for Linux*** The values in this section of the table must match those used in "Channel configuration for Linux" on page 40, as indicated.			
C	Remote queue manager name	A	**LINUX**
D	Remote queue name		**LINUX.REMOTEQ**
E	Queue name at remote system	B	**LINUX.LOCALQ**
F	Transmission queue name		**LINUX**
G	Sender (SNA) channel name		**WINNT.LINUX.SNA**
H	Sender (TCP) channel name		**WINNT.LINUX.TCP**
I	Receiver (SNA) channel name	G	**LINUX.WINNT.SNA**
J	Receiver (TCP) channel name	H	**LINUX.WINNT.TCP**
IBM i IBM i ***Connection to IBM MQ for IBM i*** The values in this section of the table must match those used in "Channel configuration for IBM i" on page 33, as indicated.			
C	Remote queue manager name	A	**AS400**
D	Remote queue name		**AS400.REMOTEQ**
E	Queue name at remote system	B	**AS400.LOCALQ**
F	Transmission queue name		**AS400**
G	Sender (SNA) channel name		**WINNT.AS400.SNA**
H	Sender (TCP) channel name		**WINNT.AS400.TCP**
I	Receiver (SNA) channel name	G	**AS400.WINNT.SNA**
J	Receiver (TCP) channel name	H	**AS400.WINNT.TCP**
z/OS z/OS ***Connection to IBM MQ for z/OS*** The values in this section of the table must match those used in "Channel configuration for z/OS" on page 57, as indicated.			
C	Remote queue manager name	A	**MVS**
D	Remote queue name		**MVS.REMOTEQ**
E	Queue name at remote system	B	**MVS.LOCALQ**
F	Transmission queue name		**MVS**
G	Sender (SNA) channel name		**WINNT.MVS.SNA**
H	Sender (TCP) channel name		**WINNT.MVS.TCP**
I	Receiver (SNA) channel name	G	**MVS.WINNT.SNA**
J	Receiver (TCP/IP) channel name	H	**MVS.WINNT.TCP**

Table 7. Configuration examples for IBM MQ for Windows (continued)			
	Parameter Name	**Reference**	**Example Used**
z/OS z/OS ***Connection to IBM MQ for z/OS using queue sharing groups*** The values in this section of the table must match those used in "Shared channel configuration example" on page 66, as indicated.			
C	Remote queue manager name	A	**QSG**
D	Remote queue name		**QSG.REMOTEQ**
E	Queue name at remote system	B	**QSG.SHAREDQ**
F	Transmission queue name		**QSG**
G	Sender (SNA) channel name		**WINNT.QSG.SNA**
H	Sender (TCP) channel name		**WINNT.QSG.TCP**
I	Receiver (SNA) channel name	G	**QSG.WINNT.SNA**
J	Receiver (TCP/IP) channel name	H	**QSG.WINNT.TCP**

Windows *IBM MQ for Windows sender-channel definitions using SNA*

A code sample.

```
def ql (AIX) +                                F
    usage(xmitq) +
    replace

def qr (AIX.REMOTEQ) +                        D
    rname(AIX.LOCALQ) +                       E
    rqmname(AIX) +                            C
    xmitq(AIX) +                              F
    replace

def chl (WINNT.AIX.SNA) chltype(sdr) +        G
    trptype(lu62) +
    conname(AIXCPIC) +                        18
    xmitq(AIX) +                              F
    replace
```

Windows *IBM MQ for Windows receiver-channel definitions using SNA*

A code sample.

```
def ql (WINNT.LOCALQ) replace                 B

def chl (AIX.WINNT.SNA) chltype(rcvr) +       I
    trptype(lu62) +
    replace
```

Windows *IBM MQ for Windows sender-channel definitions using TCP/IP*

A code sample.

```
def ql (AIX) +                                F
    usage(xmitq) +
    replace

def qr (AIX.REMOTEQ) +                        D
    rname(AIX.LOCALQ) +                       E
    rqmname(AIX) +                            C
    xmitq(AIX) +                              F
    replace
```

```
def chl (WINNT.AIX.TCP) chltype(sdr) +          H
    trptype(tcp) +
    conname(remote_tcpip_hostname) +
    xmitq(AIX) +                                F
    replace
```

Windows *IBM MQ for Windows receiver-channel definitions using TCP*

A code sample.

```
def ql (WINNT.LOCALQ) replace                   B

def chl (AIX.WINNT.TCP) chltype(rcvr) +         J
    trptype(tcp) +
    replace
```

Windows *Automatic startup*

IBM MQ for Windows allows you to automate the startup of a queue manager and its channel initiator, channels, listeners, and command servers.

Use the IBM MQ Services snap-in to define the services for the queue manager. When you have successfully completed testing of your communications setup, set the relevant services to **automatic** within the snap-in. This file can be read by the supplied IBM MQ service when the system is started.

For more information, see Administering IBM MQ .

Windows *Running channels as processes or threads*

IBM MQ for Windows provides the flexibility to run sending channels as Windows processes or Windows threads. This is specified in the MCATYPE parameter on the sender channel definition.

Most installations run their sending channels as threads, because the virtual and real memory required to support many concurrent channel connections is reduced. However, a NetBIOS connection needs a separate process for the sending Message Channel Agent.

Example MQ configuration for z/OS

This section gives an example of how to set up communication links from IBM MQ for z/OS to IBM MQ products on other platforms.

These are the other platforms covered by this example:

- Windows
- AIX
- Compaq Tru64 UNIX
- HP-UX
- Solaris
- Linux
- IBM i
- VSE/ESA

You can also connect any of the following:

- z/OS to z/OS
- z/OS to MVS
- MVS to MVS

See “Example IBM MQ configuration for all platforms” on page 5 for background information about this section and how to use it.

z/OS Establishing a connection

To establish a connection there are a number of things to configure.

Establishing an LU 6.2 connection

For the latest information about configuring SNA over TCP/IP, refer to the following online IBM documentation: Communications Server for z/OS .

Establishing a TCP connection

Alter the queue manager object to use the correct distributed queuing parameters using the following command. You must add the name of the TCP address space to the TCPNAME queue manager attribute.

```
ALTER QMGR TCPNAME(TCPIP)
```

The TCP connection is now established. You are ready to complete the configuration.

IBM MQ for z/OS configuration

The following steps outline how to configure IBM MQ; starting and configuring channels and listeners.

1. Start the channel initiator using the command:

   ```
   /cpf   START CHINIT  1
   ```

2. Start an LU 6.2 listener using the command:

   ```
   /cpf   START LSTR LUNAME( M1 ) TRPTYPE(LU62)
   ```

   The LUNAME of M1 refers to the symbolic name you gave your LU (5). You must specify TRPTYPE(LU62), otherwise the listener assumes that you want TCP.

3. Start a TCP listener using the command:

   ```
   /cpf   START LSTR
   ```

   If you want to use a port other than 1414 (the default IBM MQ port), use the command:

   ```
   /cpf   START LSTR PORT( 1555 )
   ```

IBM MQ channels do not initialize successfully if the channel negotiation detects that the message sequence number is different at each end. You might need to reset these channels manually.

Channel configuration for z/OS

To implement the example channels, there is some configuration necessary on the z/OS queue manager.

The following sections detail the configuration to be performed on the z/OS queue manager to implement the channel described in “Example IBM MQ configuration for all platforms” on page 5.

Examples are given for connecting IBM MQ for z/OS and IBM MQ for Windows. To connect to IBM MQ on another platform use the appropriate set of values from the table in place of the values for Windows.

Note: The words in **bold** are suggested values and reflect the names of IBM MQ objects used throughout these examples. You can change them in your product installation but, if you do, make sure that you use your own values when working through the examples in this section

Table 8. Configuration examples for IBM MQ for z/OS			
ID	**Parameter Name**	**Reference**	**Example Used**
Definition for local node			
A	Queue Manager Name		**MVS**
B	Local queue name		**MVS.LOCALQ**
Windows Windows ***Connection to IBM MQ for Windows*** The values in this section of the table must match the values used in "Channel configuration for Windows" on page 52, as indicated.			
C	Remote queue manager name	A	**WINNT**
D	Remote queue name		**WINNT.REMOTEQ**
E	Queue name at remote system	B	**WINNT.LOCALQ**
F	Transmission queue name		**WINNT**
G	Sender (LU 6.2) channel name		**MVS.WINNT.SNA**
H	Sender (TCP) channel name		**MVS.WINNT.TCP**
I	Receiver (LU 6.2) channel name	G	**WINNT.MVS.SNA**
J	Receiver (TCP/IP) channel name	H	**WINNT.MVS.TCP**
AIX AIX ***Connection to IBM MQ for AIX*** The values in this section of the table must match the values used in "Channel configuration for AIX" on page 11, as indicated.			
C	Remote queue manager name	A	**AIX**
D	Remote queue name		**AIX.REMOTEQ**
E	Queue name at remote system	B	**AIX.LOCALQ**
F	Transmission queue name		**AIX**
G	Sender (LU 6.2) channel name		**MVS.AIX.SNA**
H	Sender (TCP/IP) channel name		**MVS.AIX.TCP**
I	Receiver (LU 6.2) channel name	G	**AIX.MVS.SNA**
J	Receiver (TCP/IP) channel name	H	**AIX.MVS.TCP**
HP-UX HP-UX ***Connection to IBM MQ for HP-UX*** The values in this section of the table must match the values used in "Channel configuration for HP-UX" on page 17, as indicated.			
C	Remote queue manager name	A	**HPUX**
D	Remote queue name		**HPUX.REMOTEQ**
E	Queue name at remote system	B	**HPUX.LOCALQ**
F	Transmission queue name		**HPUX**
G	Sender (LU 6.2) channel name		**MVS.HPUX.SNA**
H	Sender (TCP) channel name		**MVS.HPUX.TCP**

Table 8. Configuration examples for IBM MQ for z/OS (continued)			
ID	**Parameter Name**	**Reference**	**Example Used**
I	Receiver (LU 6.2) channel name	G	**HPUX.MVS.SNA**
J	Receiver (TCP) channel name	H	**HPUX.MVS.TCP**
Solaris Solaris ***Connection to IBM MQ for Solaris*** The values in this section of the table must match the values used in "Channel configuration for Solaris" on page 45, as indicated.			
C	Remote queue manager name	A	**SOLARIS**
D	Remote queue name		**SOLARIS.REMOTEQ**
E	Queue name at remote system	B	**SOLARIS.LOCALQ**
F	Transmission queue name		**SOLARIS**
G	Sender (LU 6.2) channel name		**MVS.SOLARIS.SNA**
H	Sender (TCP) channel name		**MVS.SOLARIS.TCP**
I	Receiver (LU 6.2) channel name	G	**SOLARIS.MVS.SNA**
J	Receiver (TCP/IP) channel name	H	**SOLARIS.MVS.TCP**
Linux Linux ***Connection to IBM MQ for Linux*** The values in this section of the table must match the values used in "Channel configuration for Linux" on page 40, as indicated.			
C	Remote queue manager name	A	**LINUX**
D	Remote queue name		**LINUX.REMOTEQ**
E	Queue name at remote system	B	**LINUX.LOCALQ**
F	Transmission queue name		**LINUX**
G	Sender (LU 6.2) channel name		**MVS.LINUX.SNA**
H	Sender (TCP) channel name		**MVS.LINUX.TCP**
I	Receiver (LU 6.2) channel name	G	**LINUX.MVS.SNA**
J	Receiver (TCP/IP) channel name	H	**LINUX.MVS.TCP**
IBM i IBM i ***Connection to IBM MQ for IBM i*** The values in this section of the table must match the values used in "Channel configuration for IBM i" on page 33, as indicated.			
C	Remote queue manager name	A	**AS400**
D	Remote queue name		**AS400.REMOTEQ**
E	Queue name at remote system	B	**AS400.LOCALQ**
F	Transmission queue name		**AS400**
G	Sender (LU 6.2) channel name		**MVS.AS400.SNA**
H	Sender (TCP/IP) channel name		**MVS.AS400.TCP**
I	Receiver (LU 6.2) channel name	G	**AS400.MVS.SNA**

Table 8. Configuration examples for IBM MQ for z/OS (continued)			
ID	**Parameter Name**	**Reference**	**Example Used**
J	Receiver (TCP/IP) channel name	H	**AS400.MVS.TCP**

IBM MQ for z/OS sender-channel definitions

This topic details the sender-channel definitions required to configure IBM MQ for z/OS using LU 6.2 or TCP.

For LU 6.2:

```
     Local Queue
            Object type :   QLOCAL
                   Name :   WINNT              F
                  Usage :   X (XmitQ)

     Remote Queue
            Object type :   QREMOTE
                   Name :   WINNT.REMOTEQ      D
  Name on remote system :   WINNT.LOCALQ       E
     Remote system name :   WINNT              C
     Transmission queue :   WINNT              F

     Sender Channel
           Channel name :   MVS.WINNT.SNA      G
         Transport type :   L (LU6.2)
Transmission queue name :   WINNT              F
        Connection name :   M3                13
```

For TCP:

```
     Local Queue
            Object type :   QLOCAL
                   Name :   WINNT              F
                  Usage :   X (XmitQ)

     Remote Queue
            Object type :   QREMOTE
                   Name :   WINNT.REMOTEQ      D
  Name on remote system :   WINNT.LOCALQ       E
     Remote system name :   WINNT              C
     Transmission queue :   WINNT              F

     Sender Channel
           Channel name :   MVS.WINNT.TCP      H
         Transport type :   T (TCP)
Transmission queue name :   WINNT              F
        Connection name :   winnt.tcpip.hostname
```

IBM MQ for z/OS receiver-channel definitions

This topic details the receiver-channel definitions required to configure IBM MQ for z/OS using LU6.2 or TCP.

For LU 6.2:

```
     Local Queue
            Object type :   QLOCAL
                   Name :   MVS.LOCALQ        B
                  Usage :   N (Normal)

     Receiver Channel
           Channel name :   WINNT.MVS.SNA     I
```

For TCP:

```
     Local Queue
            Object type :   QLOCAL
```

```
                    Name :   MVS.LOCALQ        B
                   Usage :   N (Normal)

     Receiver Channel
           Channel name :   WINNT.MVS.TCP     J
```

z/OS Example MQ configuration for z/OS using QSGs

This section gives an example of how to set up communication links to a queue-sharing group (QSG) from IBM MQ products on Windows and AIX. You can also connect from z/OS to z/OS.

Setting up communication links from a queue sharing group to a platform other than z/OS is the same as described in "Example MQ configuration for z/OS" on page 56. There are examples to other platforms in that section.

When the connection is established, you must define some channels to complete the configuration. This process is described in "IBM MQ for z/OS shared channel configuration" on page 65.

See "Example IBM MQ configuration for all platforms" on page 5 for background information about this section and how to use it.

z/OS Configuration parameters for an LU 6.2 connection

The following worksheet lists all the parameters required to set up communication from a z/OS system to one of the other IBM MQ platforms. The worksheet shows examples of the parameters, which have been tested in a working environment, and leaves space for you to enter your own values.

The steps required to set up an LU 6.2 connection are described in "Establishing an LU 6.2 connection into a queue sharing group" on page 63, with numbered cross-references to the parameters in the example.

Numbers in the Reference column indicate that the value must match that in the appropriate example elsewhere in this section. The examples that follow in this section refer to the values in the ID column. The entries in the Parameter Name column are explained in "Explanation of terms" on page 62.

Table 9. Configuration examples for z/OS using LU 6.2

ID	Parameter Name	Reference	Example Used
Definition for local node using generic resources			
1	Command prefix		**/cpf**
2	Network ID		**NETID**
3	Node name		**MVSPU**
6	Modename		**#INTER**
7	Local Transaction Program name		**MQSERIES**
8	LAN destination address		**400074511092**
9	Local LU name		**MVSLU1**
10	Generic resource name		**MVSGR**
11	Symbolic destination		**G1**
12	Symbolic destination for generic resource name		**G2**
Windows Windows ***Connection to a Windows system***			
13	Symbolic destination		**M3**
14	Modename	21	**#INTER**
15	Remote Transaction Program name	7	**MQSERIES**

Table 9. Configuration examples for z/OS using LU 6.2 (continued)			
ID	**Parameter Name**	**Reference**	**Example Used**
16	Partner LU name	5	**WINNTLU**
21	Remote node ID	4	**05D 30F65**
AIX AIX ***Connection to an AIX system***			
13	Symbolic Destination		**M4**
14	Modename	18	**#INTER**
15	Remote Transaction Program name	6	**MQSERIES**
16	Partner LU name	4	**AIXLU**

z/OS *Explanation of terms*

An explanation of the terms used in the configuration worksheet.

1 Command prefix
This term is the unique command prefix of your IBM MQ for z/OS queue manager subsystem. The z/OS system programmer defines this value at installation time, in SYS1.PARMLIB(IEFSSNss), and can tell you the value.

2 Network ID
The VTAM startup procedure in your installation is partly customized by the ATCSTRxx member of the data set referenced by the DDNAME VTAMLST. The Network ID is the value specified for the NETID parameter in this member. For Network ID, you must specify the name of the NETID that owns the IBM MQ communications subsystem. Your network administrator can tell you the value.

3 Node name
VTAM, being a low-entry network node, does not have a Control Point name for Advanced Peer-to-Peer Networking (APPN) use. It does however have a system services control point name (SSCPNAME). For node name, you must specify the name of the SSCP that owns the IBM MQ communications subsystem. This value is defined in the same ATCSTRxx member as the Network ID. Your network administrator can tell you the value.

9 Local LU name
A logical unit (LU) is software that serves as an interface or translator between a transaction program and the network. It manages the exchange of data between transaction programs. The local LU name is the unique VTAM APPLID of this IBM MQ subsystem. Your network administrator can tell you this value.

11 12 13 Symbolic destination
This term is the name you give to the CPI-C side information profile. You need a side information entry for each LU 6.2 listener.

6 14 Modename
This term is the name given to the set of parameters that control the LU 6.2 conversation. An entry with this name and similar attributes must be defined at each end of the session. In VTAM, this corresponds to a mode table entry. You network administrator can assign this table entry to you.

7 15 Transaction Program name
IBM MQ applications trying to converse with this queue manager specify a symbolic name for the program to be run at the receiving end. This has been specified in the TPNAME attribute on the channel definition at the sender. For simplicity, wherever possible use a transaction program name of MQSERIES, or in the case of a connection to VSE/ESA, where the length is limited to 4 bytes, use MQTP.

See Defining an LU6.2 connection for z/OS using APPC/MVS for more information.

8 LAN destination address
This term is the LAN destination address that your partner nodes use to communicate with this host. When you are using a 3745 network controller, it is the value specified in the LOCADD parameter for the line definition to which your partner is physically connected. If your partner nodes use other devices such as 317X or 6611 devices, the address is set during the customization of those devices. Your network administrator can tell you this value.

10 Generic resource name
A generic resource name is a unique name assigned to a group of LU names used by the channel initiators in a queue sharing group.

16 Partner LU name
This term is the LU name of the IBM MQ queue manager on the system with which you are setting up communication. This value is specified in the side information entry for the remote partner.

21 Remote node ID
For a connection to Windows, this ID is the ID of the local node on the Windows system with which you are setting up communication.

z/OS Establishing an LU 6.2 connection into a queue sharing group

There are two steps to establish an LU 6.2 connection. Defining yourself to the network and defining a connection to the partner.

z/OS *Defining yourself to the network using generic resources*

You can use VTAM Generic Resources to have one connection name to connect to the queue sharing group.

1. SYS1.PARMLIB(APPCPMxx) contains the start-up parameters for APPC. You must add a line to this file to tell APPC where to locate the sideinfo. This line must be of the form:

   ```
   SIDEINFO
       DATASET(APPC.APPCSI)
   ```

2. Add another line to SYS1.PARMLIB(APPCPMxx) to define the local LU name you intend to use for the IBM MQ LU 6.2 group listener. The line you add must take the form

   ```
   LUADD ACBNAME(mvslu1)
         NOSCHED
         TPDATA(csq.appctp)
         GRNAME(mvsgr)
   ```

   Specify values for `ACBNAME` (9), `TPDATA` and `GRNAME(10)`.

   The NOSCHED parameter tells APPC that our new LU is not using the LU 6.2 scheduler (ASCH), but has one of its own. TPDATA refers to the Transaction Program data set in which LU 6.2 stores information about transaction programs. Again, IBM MQ does not use this parameter, but it is required by the syntax of the LUADD command.

3. Start the APPC subsystem with the command:

   ```
   START APPC,SUB=MSTR,APPC=xx
   ```

   where *xx* is the suffix of the PARMLIB member in which you added the LU in step 1.

   Note: If APPC is already running, it can be refreshed with the command:

   ```
   SET APPC=xx
   ```

   The effect of this is cumulative, that is, APPC does not lose its knowledge of objects already defined to it in this member or another PARMLIB member.

4. Add the new LU to a suitable VTAM major node definition. These are typically in SYS1.VTAMLST. The APPL definition will look like the sample shown.

```
        MVSLU APPL  ACBNAME=MVSLU1,     9
                    APPXC=YES,
                    AUTOSES=0,
                    DDRAINL=NALLOW,
                    DLOGMOD=#INTER,     6
                    DMINWML=10,
                    DMINWNR=10,
                    DRESPL=NALLOW,
                    DSESLIM=60,
                    LMDENT=19,
                    MODETAB=MTCICS,
                    PARSESS=YES,
                    VERIFY=NONE,
                    SECACPT=ALREADYV,
                    SRBEXIT=YES
```

5. Activate the major node. This activation can be done with the command:

```
V,NET,ACT,majornode
```

6. Add entries defining your LU and generic resource name to the CPI-C side information data set. Use the APPC utility program ATBSDFMU to do so. Sample JCL is in *thlqual*.SCSQPROC(CSQ4SIDE) (where *thlqual* is the target library high-level qualifier for IBM MQ data sets in your installation.)

   The entries you add will look like this example:

```
     SIADD
          DESTNAME(G1)           11
          MODENAME(#INTER)
          TPNAME(MQSERIES)
          PARTNER_LU(MVSLU1)     9
     SIADD
          DESTNAME(G2)           12
          MODENAME(#INTER)
          TPNAME(MQSERIES)
          PARTNER_LU(MVSGR)      10
```

7. Alter the queue manager object to use the correct distributed queuing parameters using the following command. You must specify the local LU (9) assigned to your queue manager in the LUGROUP attribute of the queue manager.

```
ALTER QMGR LUGROUP(MVSLU1)
```

z/OS *Defining a connection to a partner*

You can define a connection to a partner by adding an entry to the CPI-C side information data set.

Note: This example is for a connection to a Windows system but the task is the same for other platforms.

Add an entry to the CPI-C side information data set to define the connection. Sample JCL to do this definition is in *thlqual*.SCSQPROC(CSQ4SIDE).

The entry you add will look like this:

```
    SIADD
         DESTNAME(M3)           13
         MODENAME(#INTER)       14
         TPNAME(MQSERIES)       15
         PARTNER_LU(WINNTLU)    16
```

z/OS *What next?*

The connection is now established. You are ready to complete the configuration.

Go to "IBM MQ for z/OS shared channel configuration" on page 65.

z/OS Establishing a TCP connection Using Sysplex Distributor

You can set up Sysplex distributor to use one connection name to connect to the queue sharing group.

1. Define a Distributed DVIPA address as follows:
   a. Add a DYNAMICXCF statement to the IPCONFIG. This statement is used for inter-image connectivity using dynamically created XCF TCP/IP links.
   b. Use the VIPADYNAMIC block on each image in the Sysplex.
      i) On the owning image, code a VIPADEFINE statement to create the DVIPA Then code a VIPADISTRIBUTE statement to distribute it to all other or selected images.
      ii) On the backup image, code a VIPABACKUP statement for the DVIPA address.
2. If more than one channel initiator will be started on any LPAR in the sysplex then add the SHAREPORT option for the port to be shared in the PORT reservation list in the PROFILE data set.

See PORT statement in the *z/OS Communications Server: IP Configuration Reference* for more information.

Sysplex Distributor balances the inbound connections between each LPAR. If there is more than one channel initiator on an LPAR, then the use of SHAREPORT passes that inbound connection to the listener port with the smallest number of connections.

When you have completed these steps, the TCP connection is established. You are ready to complete the configuration.

Go to "IBM MQ for z/OS shared channel configuration" on page 65.

z/OS IBM MQ for z/OS shared channel configuration

Configure the shared channel by starting the channel initiator and issuing appropriate commands for your configuration.

1. Start the channel initiator using the command:

```
/cpf   START CHINIT
```

2. Start an LU6.2 group listener using the command:

```
/cpf   START LSTR TRPTYPE(LU62) LUNAME( G1 ) INDISP(GROUP)
```

   The LUNAME of G1 refers to the symbolic name you gave your LU (11).
3. If you are using Virtual IP Addressing using Sysplex Distributor and want to listen on a specific address, use the command:

```
/cpf   START LSTR TRPTYPE(TCP) PORT(1555) IPADDR( mvsvipa ) INDISP(GROUP)
```

There can be only one instance of the shared channel running at a time. If you try to start a second instance of the channel it fails (the error message varies depending on other factors). The shared synchronization queue tracks the channel status.

IBM MQ channels do not initialize successfully if the channel negotiation detects that the message sequence number is different at each end. You might need to reset this manually.

z/OS *Shared channel configuration example*

To configure a shared channel, a number of steps must be completed.

The subsequent topics detail the configuration to be performed on the z/OS queue manager to implement the channel described in "Example IBM MQ configuration for all platforms" on page 5.

Examples are given for connecting IBM MQ for z/OS and Windows. To connect to IBM MQ on another platform use the appropriate set of values from the table in place of the values for Windows.

Note: The words in **bold** are suggested values and reflect the names of IBM MQ objects used throughout these examples. You can change them in your product installation but, if you do, make sure that you use your own values when working through the examples in this section.

Table 10. Configuration examples for IBM MQ for z/OS using queue sharing groups

ID	Parameter Name	Reference	Example Used
Definition for local node			
A	Queue Manager Name		**QSG**
B	Local queue name		**QSG.SHAREDQ**
Windows Windows ***Connection to IBM MQ for Windows*** The values in this section of the table must match the values used in "Channel configuration for Windows" on page 52, as indicated.			
C	Remote queue manager name	A	**WINNT**
D	Remote queue name		**WINNT.REMOTEQ**
E	Queue name at remote system	B	**WINNT.LOCALQ**
F	Transmission queue name		**WINNT**
G	Sender (LU 6.2) channel name		**QSG.WINNT.SNA**
H	Sender (TCP) channel name		**QSG.WINNT.TCP**
I	Receiver (LU 6.2) channel name	G	**WINNT.QSG.SNA**
J	Receiver (TCP/IP) channel name	H	**WINNT.QSG.TCP**
AIX AIX ***Connection to IBM MQ for AIX*** The values in this section of the table must match the values used in "Channel configuration for AIX" on page 11, as indicated.			
C	Remote queue manager name		**AIX**
D	Remote queue name		**AIX.REMOTEQ**
E	Queue name at remote system	B	**AIX.LOCALQ**
F	Transmission queue name		**AIX**
G	Sender (LU 6.2) channel name		**QSG.AIX.SNA**
H	Sender (TCP/IP) channel name		**QSG.AIX.TCP**
I	Receiver (LU 6.2) channel name	G	**AIX.QSG.SNA**
J	Receiver (TCP/IP) channel name	H	**AIX.QSG.TCP**

z/OS IBM MQ for z/OS shared sender-channel definitions

An example definition of shared sender-channels for LU 6.2 and TCP.

Using LU 6.2

```
Local Queue
            Object type :   QLOCAL
                   Name :   WINNT              F
                  Usage :   X (XmitQ)
            Disposition :   SHARED

     Remote Queue
            Object type :   QREMOTE
                   Name :   WINNT.REMOTEQ      D
  Name on remote system :   WINNT.LOCALQ       E
     Remote system name :   WINNT              C
     Transmission queue :   WINNT              F
            Disposition :   GROUP

     Sender Channel
           Channel name :   MVS.WINNT.SNA      G
         Transport type :   L (LU6.2)
Transmission queue name :   WINNT              F
        Connection name :   M3                13
            Disposition :   GROUP
```

Using TCP

```
     Local Queue
            Object type :   QLOCAL
                   Name :   WINNT              F
                  Usage :   X (XmitQ)
            Disposition :   SHARED

     Remote Queue
            Object type :   QREMOTE
                   Name :   WINNT.REMOTEQ      D
  Name on remote system :   WINNT.LOCALQ       E
     Remote system name :   WINNT              C
     Transmission queue :   WINNT              F
            Disposition :   GROUP

     Sender Channel
           Channel name :   QSG.WINNT.TCP      H
         Transport type :   T (TCP)
Transmission queue name :   WINNT              F
        Connection name :   winnt.tcpip.hostname
            Disposition :   GROUP
```

z/OS IBM MQ for z/OS shared receiver-channel definitions

An example definition of shared receiver-channels for LU 6.2 and TCP.

Using LU 6.2

```
     Local Queue
            Object type :   QLOCAL
                   Name :   QSG.SHAREDQ        B
                  Usage :   N (Normal)
            Disposition :   SHARED

     Receiver Channel
           Channel name :   WINNT.QSG.SNA      I
            Disposition :   GROUP
```

Using TCP

```
Local Queue
       Object type :   QLOCAL
              Name :   QSG.SHAREDQ        B
             Usage :   N (Normal)
       Disposition :   SHARED

Receiver Channel
      Channel name :   WINNT.QSG.TCP      J
       Disposition :   GROUP
```

z/OS Example MQ configuration for z/OS using intra-group queuing

This section describes how a typical payroll query application, that currently uses distributed queuing to transfer small messages between queue managers, could be migrated to use queue sharing groups and shared queues.

Three configurations are described to illustrate the use of distributed queuing, intra-group queuing with shared queues, and shared queues. The associated diagrams show only the flow of data in one direction, that is, from queue manager QMG1 to queue manager QMG3.

z/OS Configuration 1

Configuration 1 describes how distributed queuing is currently used to transfer messages between queue managers QMG1 and QMG3.

Configuration 1 shows a distributed queuing system that is used to transfer messages received by queue manager QMG1 from the payroll query to queue manager QMG2 and then finally on to queue manager QMG3, to be sent to the payroll server.

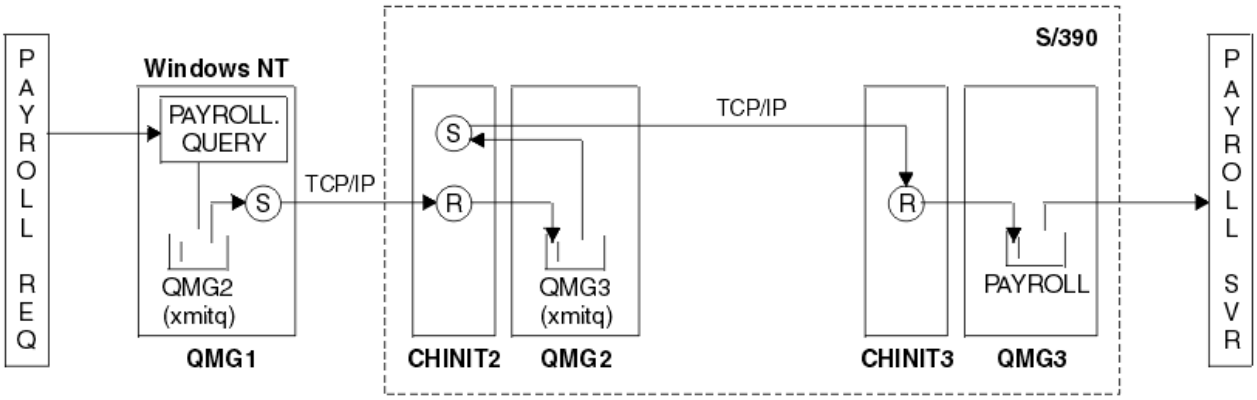

Figure 2. Configuration 1: z/OS using intra-group queuing

The flow of operations is as follows:

1. A query is entered using the payroll request application connected to queue manager QMG1.
2. The payroll request application puts the query on to remote queue PAYROLL.QUERY. As queue PAYROLL.QUERY resolves to transmission queue QMG2, the query is put on to transmission queue QMG2.
3. Sender channel (S) on queue manager QMG1 delivers the query to the partner receiver channel (R) on queue manager QMG2.
4. Receiver channel (R) on queue manager QMG2 puts the query on to queue PAYROLL on queue manager QMG3. As queue PAYROLL on QMG3 resolves to transmission queue QMG3, the query is put on to transmission queue QMG3.
5. Sender channel (S) on queue manager QMG2 delivers the query to the partner receiver channel (R) on queue manager QMG3.

6. Receiver channel (R) on queue manager QMG3 puts the query on to local queue PAYROLL.
7. The payroll server application connected to queue manager QMG3 retrieves the query from local queue PAYROLL, processes it, and generates a suitable reply.

z/OS *Configuration 1 definitions*

The definitions required for Configuration 1 are as follows (note that the definitions do not take into account triggering, and that only channel definitions for communication using TCP/IP are provided).

On QMG1

Remote queue definition:

```
DEFINE QREMOTE(PAYROLL.QUERY) DESCR('Remote queue for QMG3') REPLACE +
PUT(ENABLED) RNAME(PAYROLL) RQMNAME(QMG3) XMITQ(QMG2)
```

Transmission queue definition:

```
DEFINE QLOCAL(QMG2) DESCR('Transmission queue to QMG2') REPLACE +
PUT(ENABLED) USAGE(XMITQ) GET(ENABLED)
```

Sender channel definition (for TCP/IP):

```
DEFINE CHANNEL(QMG1.TO.QMG2) CHLTYPE(SDR) TRPTYPE(TCP) REPLACE +
DESCR('Sender channel to QMG2') XMITQ(QMG2) CONNAME('MVSQMG2(1415)')
```

Here you replace MVSQMG2(1415) with your queue manager connection name and port.

Receiver channel definition (for TCP/IP):

```
DEFINE CHANNEL(QMG2.TO.QMG1) CHLTYPE(RCVR) TRPTYPE(TCP) +
REPLACE DESCR('Receiver channel from QMG2')
```

Reply-to queue definition:

```
DEFINE QLOCAL(PAYROLL.REPLY) REPLACE PUT(ENABLED) GET(ENABLED) +
DESCR('Reply queue for replies to payroll queries sent to QMG3')
```

On QMG2

Transmission queue definition:

```
DEFINE QLOCAL(QMG1) DESCR('Transmission queue to QMG1') REPLACE +
PUT(ENABLED) USAGE(XMITQ) GET(ENABLED)

DEFINE QLOCAL(QMG3) DESCR('Transmission queue to QMG3') REPLACE +
PUT(ENABLED) USAGE(XMITQ) GET(ENABLED)
```

Sender channel definitions (for TCP/IP):

```
DEFINE CHANNEL(QMG2.TO.QMG1) CHLTYPE(SDR) TRPTYPE(TCP) REPLACE +
DESCR('Sender channel to QMG1') XMITQ(QMG1) CONNAME('WINTQMG1(1414)')
```

Here you replace WINTQMG1(1414) with your queue manager connection name and port.

```
DEFINE CHANNEL(QMG2.TO.QMG3) CHLTYPE(SDR) TRPTYPE(TCP) REPLACE +
DESCR('Sender channel to QMG3') XMITQ(QMG3) CONNAME('MVSQMG3(1416)')
```

Here you replace MVSQMG3(1416) with your queue manager connection name and port.

Receiver channel definition (for TCP/IP):

```
DEFINE CHANNEL(QMG1.TO.QMG2) CHLTYPE(RCVR) TRPTYPE(TCP) +
REPLACE DESCR('Receiver channel from QMG1')

DEFINE CHANNEL(QMG3.TO.QMG2) CHLTYPE(RCVR) TRPTYPE(TCP) +
REPLACE DESCR('Receiver channel from QMG3')
```

On QMG3

Local queue definition:

```
DEFINE QLOCAL(PAYROLL) DESCR('Payroll query request queue') REPLACE +
PUT(ENABLED) USAGE(NORMAL) GET(ENABLED) SHARE

DEFINE QLOCAL(QMG2) DESCR('Transmission queue to QMG2') REPLACE +
PUT(ENABLED) USAGE(XMITQ) GET(ENABLED)
```

Sender channel definitions (for TCP/IP):

```
DEFINE CHANNEL(QMG3.TO.QMG2) CHLTYPE(SDR) TRPTYPE(TCP) REPLACE +
DESCR('Sender channel to QMG2) XMITQ(QMG2) CONNAME('MVSQMG2(1415)')
```

Here you replace MVSQMG2(1415) with your queue manager connection name and port.

Receiver channel definition (for TCP/IP):

```
DEFINE CHANNEL(QMG2.TO.QMG3) CHLTYPE(RCVR) TRPTYPE(TCP) +
REPLACE DESCR('Receiver channel from QMG2)
```

z/OS Configuration 2

Configuration 2 describes how queue sharing groups and intra-group queuing can be used, with no effect on the back-end payroll server application, to transfer messages between queue managers QMG1 and QMG3.

Configuration 2 shows a distributed queuing system that uses queue sharing groups and intra-group queuing to transfer messages from the payroll request application to the payroll server. This configuration removes the need for channel definitions between queue managers QMG2 and QMG3 because intra-group queuing is used to transfer messages between these two queue managers.

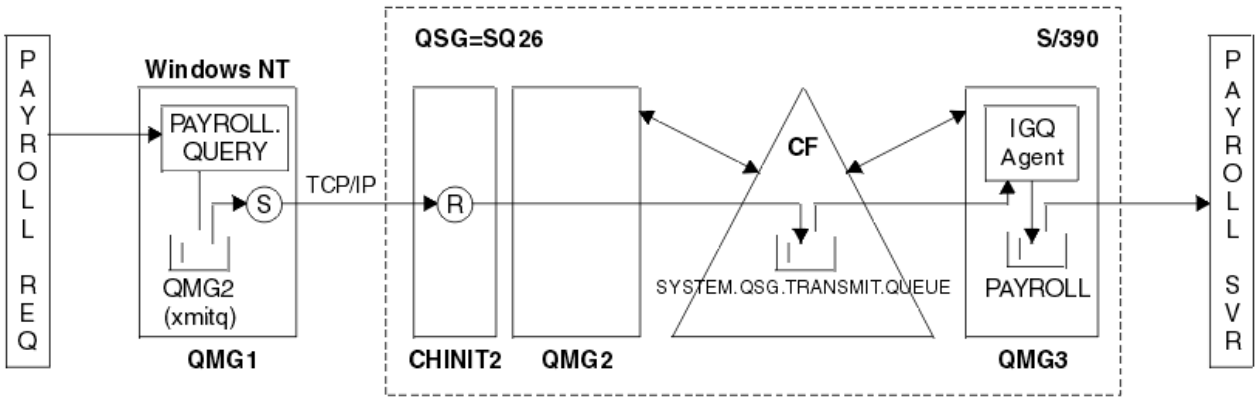

Figure 3. Configuration 2

The flow of operations is as follows:

1. A query is entered using the payroll request application connected to queue manager QMG1.
2. The payroll request application puts the query on to remote queue PAYROLL.QUERY. As queue PAYROLL.QUERY resolves to transmission queue QMG2, the query is put on to transmission queue QMG2.
3. Sender channel (S) on queue manager QMG1 delivers the query to the partner receiver channel (R) on queue manager QMG2.
4. Receiver channel (R) on queue manager QMG2 puts the query on to queue PAYROLL on queue manager QMG3. As queue PAYROLL on QMG3 resolves to shared transmission queue SYSTEM.QSG.TRANSMIT.QUEUE, the query is put on to shared transmission queue SYSTEM.QSG.TRANSMIT.QUEUE.
5. IGQ agent on queue manager QMG3 retrieves the query from shared transmission queue SYSTEM.QSG.TRANSMIT.QUEUE, and puts it on to local queue PAYROLL on queue manager QMG3.
6. The payroll server application connected to queue manager QMG3 retrieves the query from local queue PAYROLL, processes it, and generates a suitable reply.

Note: The payroll query example transfers small messages only. If you need to transfer both persistent and non-persistent messages, a combination of Configuration 1 and Configuration 2 can be established, so that large messages can be transferred using the distributed queuing route, while small messages can be transferred using the potentially faster intra-group queuing route.

z/OS *Configuration 2 definitions*

The definitions required for Configuration 2 are as follows (note that the definitions do not take into account triggering, and that only channel definitions for communication using TCP/IP are provided).

It is assumed that queue managers QMG2 and QMG3 are already configured to be members of the same queue sharing group.

On QMG1

Remote queue definition:

```
DEFINE QREMOTE(PAYROLL.QUERY) DESCR('Remote queue for QMG3') REPLACE +
PUT(ENABLED) RNAME(PAYROLL) RQMNAME(QMG3) XMITQ(QMG2)
```

Transmission queue definition:

```
DEFINE QLOCAL(QMG2) DESCR('Transmission queue to QMG2') REPLACE +
PUT(ENABLED) USAGE(XMITQ) GET(ENABLED)
```

Sender channel definition (for TCP/IP):

```
DEFINE CHANNEL(QMG1.TO.QMG2) CHLTYPE(SDR) TRPTYPE(TCP) REPLACE +
DESCR('Sender channel to QMG2') XMITQ(QMG2) CONNAME('MVSQMG2(1415)')
```

Here you replace MVSQMG2(1415) with your queue manager connection name and port.

Receiver channel definition (for TCP/IP):

```
DEFINE CHANNEL(QMG2.TO.QMG1) CHLTYPE(RCVR) TRPTYPE(TCP) +
REPLACE DESCR('Receiver channel from QMG2')
```

Reply-to queue definition:

```
DEFINE QLOCAL(PAYROLL.REPLY) REPLACE PUT(ENABLED) GET(ENABLED) +
DESCR('Reply queue for replies to payroll queries sent to QMG3')
```

On QMG2

Transmission queue definition:

```
DEFINE QLOCAL(QMG1) DESCR('Transmission queue to QMG1') REPLACE +
PUT(ENABLED) USAGE(XMITQ) GET(ENABLED)

DEFINE QLOCAL(SYSTEM.QSG.TRANSMIT.QUEUE) QSGDISP(SHARED) +
DESCR('IGQ Transmission queue') REPLACE PUT(ENABLED) USAGE(XMITQ) +
GET(ENABLED) INDXTYPE(CORRELID) CFSTRUCT('APPLICATION1') +
DEFSOPT(SHARED) DEFPSIST(NO)
```

Here you replace APPLICATION1 with your defined CF structure name. Also note that this queue, being a shared queue, need only be defined on one of the queue managers in the queue sharing group.

Sender channel definitions (for TCP/IP):

```
DEFINE CHANNEL(QMG2.TO.QMG1) CHLTYPE(SDR) TRPTYPE(TCP) REPLACE +
DESCR('Sender channel to QMG1') XMITQ(QMG1) CONNAME('WINTQMG1(1414)')
```

Here you replace WINTQMG1(1414) with your queue manager connection name and port.

Receiver channel definition (for TCP/IP):

```
DEFINE CHANNEL(QMG1.TO.QMG2) CHLTYPE(RCVR) TRPTYPE(TCP) +
REPLACE DESCR('Receiver channel from QMG1')
```

Queue Manager definition:

```
ALTER QMGR IGQ(ENABLED)
```

On QMG3

Local queue definition:

```
DEFINE QLOCAL(PAYROLL) DESCR('Payroll query request queue') REPLACE +
PUT(ENABLED) USAGE(NORMAL) GET(ENABLED) SHARE
```

Queue Manager definition:

```
ALTER QMGR IGQ(ENABLED)
```

z/OS Configuration 3

Configuration 3 describes how queue sharing groups and shared queues can be used, with no effect on the back-end payroll server application, to transfer messages between queue managers QMG1 and QMG3.

Configuration 3 shows a distributed queuing system that uses queue sharing groups and shared queues to transfer messages between queue manager QMG1 and queue manager QMG3.

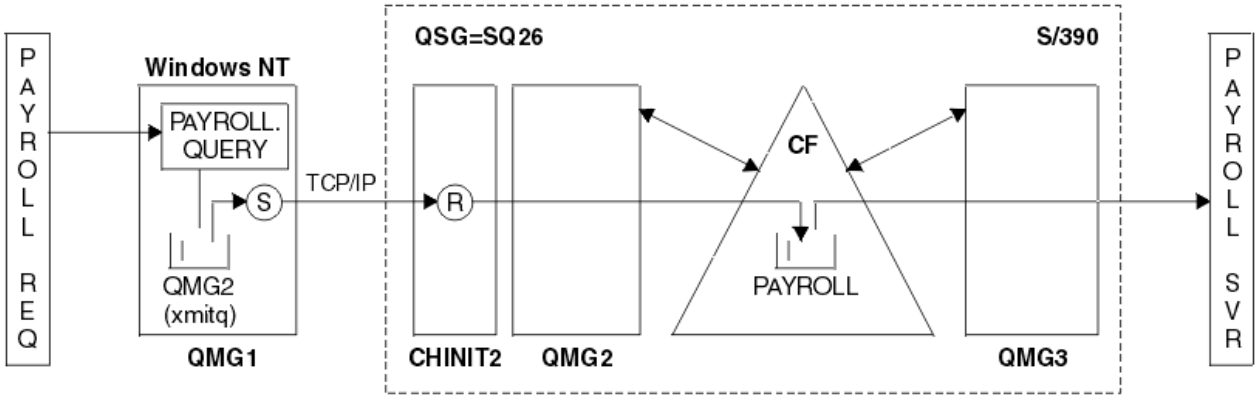

Figure 4. Configuration 3

The flow of operations is:

1. A query is entered using the payroll request application connected to queue manager QMG1.
2. The payroll request application puts the query on to remote queue PAYROLL.QUERY. As queue PAYROLL.QUERY resolves to transmission queue QMG2, the query is put on to transmission queue QMG2.
3. Sender channel (S) on queue manager QMG1 delivers the query to the partner receiver channel (R) on queue manager QMG2.
4. Receiver channel (R) on queue manager QMG2 puts the query on to shared queue PAYROLL.
5. The payroll server application connected to queue manager QMG3 retrieves the query from shared queue PAYROLL, processes it, and generates a suitable reply.

This configuration is certainly the simplest to configure. However, distributed queuing or intra-group queuing would need to be configured to transfer replies (generated by the payroll server application connected to queue manager QMG3) from queue manager QMG3 to queue manager QMG2, and then on to queue manager QMG1. (See "What the queue sharing group example for z/OS shows" on page 170 for the configuration used to transfer replies back to the payroll request application.)

No definitions are required on QMG3.

z/OS *Configuration 3 definitions*

The definitions required for Configuration 3 are as follows (note that the definitions do not take into account triggering, and that only channel definitions for communication using TCP/IP are provided).

It is assumed that queue managers QMG2 and QMG3 are already configured to be members of the same queue sharing group.

On QMG1

Remote queue definition:

```
DEFINE QREMOTE(PAYROLL.QUERY) DESCR('Remote queue for QMG3') REPLACE +
PUT(ENABLED) RNAME(PAYROLL) RQMNAME(QMG3) XMITQ(QMG2)
```

Transmission queue definition:

```
DEFINE QLOCAL(QMG2) DESCR('Transmission queue to QMG2') REPLACE +
PUT(ENABLED) USAGE(XMITQ) GET(ENABLED)
```

Sender channel definition (for TCP/IP):

```
DEFINE CHANNEL(QMG1.TO.QMG2) CHLTYPE(SDR) TRPTYPE(TCP) +
REPLACE DESCR('Sender channel to QMG2') XMITQ(QMG2) CONNAME('MVSQMG2(1415)')
```

Here you replace MVSQMG2(1415) with your queue manager connection name and port.

Receiver channel definition (for TCP/IP):

```
DEFINE CHANNEL(QMG2.TO.QMG1) CHLTYPE(RCVR) TRPTYPE(TCP) +
REPLACE DESCR('Receiver channel from QMG2')
```

Reply-to queue definition:

```
DEFINE QLOCAL(PAYROLL.REPLY) REPLACE PUT(ENABLED) GET(ENABLED) +
DESCR('Reply queue for replies to payroll queries sent to QMG3')
```

On QMG2

Transmission queue definition:

```
DEFINE QLOCAL(QMG1) DESCR('Transmission queue to QMG1') REPLACE +
PUT(ENABLED) USAGE(XMITQ) GET(ENABLED)
```

Sender channel definitions (for TCP/IP):

```
DEFINE CHANNEL(QMG2.TO.QMG1) CHLTYPE(SDR) TRPTYPE(TCP) +
REPLACE DESCR('Sender channel to QMG1') XMITQ(QMG1) CONNAME('WINTQMG1(1414)')
```

Here you replace WINTQMG1(1414) with your queue manager connection name and port.

Receiver channel definition (for TCP/IP):

```
DEFINE CHANNEL(QMG1.TO.QMG2) CHLTYPE(RCVR) TRPTYPE(TCP) +
REPLACE DESCR('Receiver channel from QMG1')
```

Local queue definition:

```
DEFINE QLOCAL(PAYROLL) QSGDISP(SHARED) DESCR('Payroll query request queue') +
REPLACE PUT(ENABLED) USAGE(NORMAL) GET(ENABLED) SHARE +
DEFSOPT(SHARED) DEFPSIST(NO) CFSTRUCT(APPLICATION1)
```

Here you replace APPLICATION1 with your defined CF structure name. Also note that this queue, being a shared queue, need only be defined on one of the queue managers in the queue sharing group.

On QMG3

No definitions are required on QMG3.

z/OS Running the example

After setting up the sample, you can run the sample.

For Configuration 1:

1. Start queue managers QMG1, QMG2, and QMG3.
2. Start channel initiators for QMG2 and QMG3.

3. Start the listeners on QMG1 to listen to port 1414, QMG2 to listen on port 1415, and QMG3 to listen on port 1416.
4. Start sender channels on QMG1, QMG2, and QMG3.
5. Start the payroll query requesting application connected to QMG1.
6. Start the payroll server application connected to QMG3.
7. Submit a payroll query request to QMG3 and wait for the payroll reply.

For Configuration 2:

1. Start queue managers QMG1, QMG2, and QMG3.
2. Start the channel initiator for QMG2.
3. Start the listeners on QMG1 to listen on port 1414, and QMG2 to listen on port 1415.
4. Start the sender channel on QMG1 and QMG2.
5. Start the payroll query requesting application connected to QMG1.
6. Start the payroll server application connected to QMG3.
7. Submit a payroll query request to QMG3 and wait for the payroll reply.

For Configuration 3:

1. Start queue managers QMG1, QMG2, and QMG3.
2. Start the channel initiator for QMG2.
3. Start the listeners on QMG1 to listen on port 1414, and QMG2 to listen on port 1415.
4. Start sender channels on QMG1 and QMG2.
5. Start the payroll query requesting application connected to QMG1.
6. Start the payroll server application connected to QMG3.
7. Submit a payroll query request to QMG3 and wait for the payroll reply.

z/OS Expanding the example

The example can be expanded in a number of ways.

The example can be:

- Expanded to use channel triggering as well as application (PAYROLL and PAYROLL.REPLY queue) triggering.
- Configured for communication using LU6.2.
- Expanded to configure more queue managers to the queue sharing group. Then the server application can be cloned to run on other queue manager instances to provide multiple servers for the PAYROLL query queue.
- Expanded to increase the number of instances of the payroll query requesting application to demonstrate the processing of requests from multiple clients.
- Expanded to use security (IGQAUT and IGQUSER).

Linux IBM i UNIX IBM MQ file system permissions applied to /var/mqm

The following information describes the security applied to the files and directories under /var/mqm/ and why the file-system permissions are set as they are. In order to ensure the correct operation of IBM MQ you should not alter the file system permissions as set by IBM MQ

crtmqdir command

From IBM MQ 9.0.3, if your enterprise has changed any of the /var/mqm file permissions, for whatever reason, you can update the permissions, or add directories, by using the **crtmqdir** command

IBM MQ file system Security on UNIX, Linux, and IBM i

The files under the IBM MQ data directory (/var/mqm) are used to store:

- IBM MQ configuration data
- Application data (IBM MQ objects and the data contained within IBM MQ messages)
- Run-time control information
- Monitoring information (messages and FFST files)

Access to this data is controlled using file system permissions with some of the data being accessible to all users while other data is restricted only to members of the IBM MQ Administrator group 'mqm' (or QMQM on IBM i).

Access is granted in the following three categories:

mqm group only

The files and directories in this category are only accessible to IBM MQ Administrators (members of the 'mqm' group) and the IBM MQ queue manager processes.

The file permissions for these files and directories are:

```
-rwxrwx---    mqm:mqm      (UNIX and Linux)
-rwxrwx---    QMQMADM:QMQM (IBM i)
```

An example of the files and directories in this category is:

```
/var/mqm/qmgrs/QMGR/qm.ini
/var/mqm/qmgrs/QMGR/channel/
/var/mqm/qmgrs/QMGR/channel/SYSTEM!DEF!SCRVONN
/var/mqm/qmgrs/QMGR/queues/
/var/mqm/qmgrs/QMGR/queues/SYSTEM!DEFAULT!LOCAL!QUEUES/
/var/mqm/qmgrs/QMGR/errors/
/var/mqm/qmgrs/QMGR/errors/AMQERR01.LOG
/var/mqm/qmgrs/QMGR/ssl/
/var/mqm/qmgrs/QMGR/@qmgr/
/var/mqm/qmgrs/QMGR/@qmpersist/
...
```

All users read access - mqm group members read and write access

The files and directories in this category can be read by all users, but only members of the 'mqm' group can modify these files and manipulate these directories.

The file permissions for these files and directories are:

```
-rwxrwxr-x    mqm:mqm      (UNIX and Linux)
-rwxrwxr-x    QMQMADM:QMQM (IBM i)
```

An example of the files and directories in this category is:

```
/var/mqm/mqs.ini
/var/mqm/exits/
/var/mqm/qmgrs/
/var/mqm/qmgrs/QMGR/
/var/mqm/qmgrs/QMGR/@app/
/var/mqm/qmgrs/QMGR/@ipcc/
```

Attention: You should only set execute permissions on executable files and scripts. For example, on Linux when the **crtmqm** command runs, the following file permissions are set:

```
-rw-rw---- mqm mqm /var/mqm/qmgrs/QMGR/qm.ini
-rw-rw---- mqm mqm /var/mqm/qmgrs/QMGR/channel/SYSTEM!DEF!SCRVONN
-rw-rw---- mqm mqm /var/mqm/qmgrs/QMGR/errors/AMQERR01.LOG
-rw-rw-r-- mqm mqm /var/mqm/mqs.ini
```

IBM MQ 8.0:

```
/var/mqm/sockets/@SYSTEM
/var/mqm/sockets/QMGR/@app/hostname
/var/mqm/sockets/QMGR/@ipcc/hostname
```

All users read and write access

Files that have read and write access for all users

IBM MQ has no *regular* files that have world writable file permissions (777). However there are a number of *special* files that appear as having world writable file permissions.

These special files provide no security exposure. Although the permissions are shown as 777, they are not *regular* files and you cannot write directly to them.

These special files are:

Symbolic links

Symbolic links are identified by the 'l' character at the start of their permissions. The permissions on the symbolic link have no effect on who is able to access the target file, as access to the command is controlled by the permissions on the target of the symbolic link.

On most UNIX and Linux systems it is not possible to change the permissions on symbolic links, so they always appear as `lrwxrwxrwx`.

Socket files

Socket files are special files created by the operating system, as a result of a process creating a UNIX domain socket. These files can be identified by the 's' at the start of the file permissions, that is `srwxrwxrwx`.

The permissions on the file do not grant access to the file itself, but define who can connect to the UNIX domain socket.

IBM MQ uses a number of these socket files and the permissions are always set according to who is allowed to communicate with the socket.

The following directories contain socket files that have read/write permissions for all users (`srwxrwxrwx`).

IBM MQ 8.0:

```
/var/mqm/sockets/QMGR/zsocketEC/hostname/Zsocket_*
```

Socket files used by applications that connect to IBM MQ using isolated bindings.

```
/var/mqm/sockets/QMGR/@ipcc/ssem/hostname/*
```

Directories that have read and write access for all users

There are times when IBM MQ applications need to create files under the IBM MQ data directory. To ensure that applications are able to create files when they are required, a number of directories are granted world write access, which means that any user on the system can create files within that directory.

With the exception of the errors logs files, that can be written to by any member of the 'mqm' group, all files created in these directories are created with restricted permissions that allows only the file creator write access. This allows the system administrator to track the user ID of all data written to files in these directories.

/var/mqm/errors/

This directory contains the system error log files and FFST files. The permission of this directory is 'drwxrwsrwt' meaning that all users on the system can create files in this directory.

The SetGroupId bit 's' indicates that all files created in this directory have the group ownership of 'mqm'.

The 't' sticky bit is not set by default on this directory, but an IBM MQ administrator can set this explicitly, to allow users to delete only the files that they create.

Note: IBM i This feature is not available on IBM i.

AMQERR0*.LOG

These error log files can only be written to directly by members of the group but any user can read the messages written to these files (permission: -rw-rw-r--).

AMQnnnnn.*.FDC

These files contain FFST information written when an error occurs in the queue manager or in an application written by a user. These files are created with the permissions -rw-r-----.

/var/mqm/trace/

Trace files are written to this directory when IBM MQ trace is enabled. IBM MQ trace is written by all process associated with a queue manager for which trace is enabled.

The permissions of this directory are 'drwxrwsrwt' meaning that all users on the system can create files in this directory.

The SetGroupId bit 's' indicates that all files created in this directory have the group ownership of 'mqm'.

The 't' sticky bit is not set by default on this directory, but an IBM MQ administrator can set this explicitly, to allow users to delete only the files that they create.

Note: IBM i This feature is not available on IBM i.

AMQnnnnn.*.TRC

These files contain the trace data written by each process which is tracing and are created with permissions -rw-r-----

The permissions on this directory are drwxrwsrwt and the permissions of the socket files created in this directory are srwx------.

IBM MQ 8.0:

```
/var/mqm/sockets/QMGR/zsocketapp/hostname/
```

This directory is used by applications that connect to the IBM MQ queue manager using *isolated* bindings. During connect processing a socket file is created by the connecting application in this directory. The socket file is removed after the connection is made to the queue manager.

The permissions on this directory are drwxrwsrwt and the permissions of the socket files created in this directory are srwx------.

The SetGroupId bit 's' on this directory ensures that all files created in this directory have the group ownership of 'mqm'.

On all platforms except IBM i, this directories also has the 't' sticky bit set which prevents a user from deleting any files except the ones for which they are the owner. This prevents an unauthorized user from deleting files that they do not own.

```
/var/mqm/sockets/QMGR/@ipcc/ssem/hostname/
/var/mqm/sockets/QMGR/@app/ssem/hostname/
```

UNIX For processes that connect to IBM MQ using *shared* bindings then UNIX domain sockets might be used to synchronize between the application and the queue manager. When UNIX domain sockets are being used then the associated socket file is created in these directories.

The permissions on these directories are `drwxrwsrwt` and the permissions of the socket files created in these directories are `srwxrwxrwx`.

The `SetGroupId` bit 's' on these directories ensures that all files created in these directories have the group ownership of 'mqm'.

On all platforms except IBM i, these directories also have the 't' sticky bit set which prevents a user from deleting any files except the ones for which they are the owner. This prevents an unauthorized user from deleting files that they do not own.

Use of System V IPC resources by IBM MQ

IBM MQ uses System V shared memory and semaphores for inter-process communication. These resources are grouped according to how they are used with each group having appropriate ownership and access permissions.

To verify which of the System V IPC resources on a system belong to IBM MQ you can:

- Check the ownership.

  The owning user of IBM MQ System V IPC resources is always the 'mqm' user on UNIX platforms and Linux. On IBM i the owning user is 'QMQM'.
- IBM MQ 8.0 and later, use the `amqspdbg` utility.

  The `amqspdbg` utility which is shipped with IBM MQ can be used to display the shared memory and semaphore id's for a given queue manager.

  You must issue the command once for the 'system' group of System V resources created by IBM MQ

  ```
  # amqspbg -z -I
  ```

  and then four times for each queue manager on the system to get the complete list of System V resources used by IBM MQ. Assume a queue manager name of `QMGR1` in the following examples:.

  ```
  # amqspdbg -i QMGR1 -I
  # amqspdbg -q QMGR1 -I
  # amqspdbg -p QMGR1 -I
  # amqspdbg -a QMGR1 -I
  ```

The access permissions on the System V resources created by IBM MQ are set to grant only the correct level of access to the permitted users. A number of the System V IPC resources created by IBM MQ are accessible to all users on the machine and have permissions of `-rw-rw-rw-`.

The **-g** *ApplicationGroup* parameter on the crtmqm command can be used to restrict access to a queue manager to membership of a specific operating system group. The use of this restricted group functionality restricts the permissions granted on the System V IPC resources further.

Linux UNIX IBM MQ file permissions in /opt/mqm with setuid for mqm

The following information covers the situation where your security team has flagged some of the executable IBM MQ files in the directory tree $MQ_INSTALLATION_PATH, in violation of local security policies. The default location in AIX is /usr/mqm and for the other UNIX operating systems is /opt/mqm. If you have installed IBM MQ in a non-default directory, such as /opt/mqm90, or if you have multiple installations, the details in this topic still apply.

Cause of the problem

Your security team has identified the following areas of concern under $*MQ_INSTALLATION_PATH*:

1. Files in /opt/mqm/bin directory are setuid for the owner of the directory tree where they reside. For example:

```
dr-xr-xr-x   mqm  mqm   ${MQ_INSTALLATION_PATH}/bin
-r-sr-s---   mqm  mqm   ${MQ_INSTALLATION_PATH}/bin/addmqinf
-r-sr-s---   mqm  mqm   ${MQ_INSTALLATION_PATH}/bin/amqcrsta
-r-sr-s---   mqm  mqm   ${MQ_INSTALLATION_PATH}/bin/amqfcxba
...
```

2. Practically all the directories and files are owned by "mqm:mqm" except for the following, which are owned by root:

```
dr-xr-x---   root mqm   ${MQ_INSTALLATION_PATH}/bin/security
-r-sr-x---   root mqm   ${MQ_INSTALLATION_PATH}/bin/security/amqoamax
-r-sr-x---   root mqm   ${MQ_INSTALLATION_PATH}/bin/security/amqoampx
```

   This subdirectory needs to be owned by root, because these are the executable files that interact with the operating system when the user from an IBM MQ client specifies a password, and this password is passed by the IBM MQ queue manager to the operating system to confirm if the password is valid or is not valid.

3. User does not own files in /opt/mqm/lib/iconv directory (this directory does not exist on AIX). For example:

```
dr-xr-xr-x   mqm  mqm   ${MQ_INSTALLATION_PATH}/lib/iconv
-r--r--r--   bin  bin   ${MQ_INSTALLATION_PATH}/lib/iconv/002501B5.tbl
-r--r--r--   bin  bin   ${MQ_INSTALLATION_PATH}/lib/iconv/002501F4.tbl
-r--r--r--   bin  bin   ${MQ_INSTALLATION_PATH}/lib/iconv/00250333.tbl
...
```

4. The fix pack maintenance directory on RPM-based Linux systems. When fix packs are installed, the existing files are saved under this directory in a structure similar to that shown in the following example, except that in this example V.R represents the IBM MQ version and release number and the subdirectories that appear depend on the fix packs that have been installed:

```
drwx------   root root  ${MQ_INSTALLATION_PATH}/maintenance
drwxr-xr-x   root root  ${MQ_INSTALLATION_PATH}/maintenance/V.R.0.1
drwxr-xr-x   root root  ${MQ_INSTALLATION_PATH}/maintenance/V.R.0.3
drwxr-xr-x   root root  ${MQ_INSTALLATION_PATH}/maintenance/V.R.0.4
...
```

Resolving the problem

One of the concerns on UNIX systems with respect to setuid programs was that the system security could be compromised by manipulating environment variables such as LD* (LD_LIBRARY_PATH, LIBPATH on AIX, and so on). This is no longer a concern, as various UNIX operating systems now ignore these LD* environment variables when loading setuid programs.

1. Why some of the IBM MQ programs are mqm-setuid or mqm-setgid.

   In IBM MQ, the user id "mqm" and any ID which is a part of the "mqm" group are the IBM MQ administrative users.

IBM MQ queue manager resources are protected by authenticating against this user. Since the queue manager processes use and modify these queue manager resources, the queue manager processes require "mqm" authority to access the resources. Therefore, IBM MQ queue manager support processes are designed to run with the effective user-id of "mqm".

To help non-administrative users accessing IBM MQ objects, IBM MQ provides an Object Authority Manager (OAM) facility, whereby authorities can be granted and revoked on the need of the application run by the non-administrative user.

With the ability to grant different levels of authentications for users and the fact that **setuid** and **setgid** programs ignore LD* variables, the IBM MQ binary and library files do not compromise the security of your system in any way.

2. It is not possible to change the permissions to satisfy the security policy of your enterprise without jeopardizing IBM MQ functionality.

   You must not change the permissions and ownerships of any of the IBM MQ binaries and libraries. IBM MQ functionality can suffer due to this kind of change, such that queue manager processes might fail to access some of the resources.

   Note that the permissions and ownerships do not pose any security threat to the system.

   Linux hard drives/disks where IBM MQ is installed or where IBM MQ data is located must not be mounted with the nosuid option. This configuration might inhibit IBM MQ functionality.

   For more information see "IBM MQ file system permissions applied to /var/mqm" on page 76.

Related concepts

Filesystem

Windows IBM MQ file system permissions on Windows

The following information describes the security applied to the files and directories on Windows. In order to ensure the correct operation of IBM MQ you should not alter the file system permissions as set by IBM MQ.

Data directory

Note: The permissions that are set on the root of this directory, are inherited downwards throughout the directory structure.

The directories under the data directory (DATADIR) are set with the following permissions, apart from the exceptions detailed in the following text.

Administrators
: Full control

mqm group
: Full control

SYSTEM
: Full control

Everyone
: Read and execute

The exceptions are:

DATADIR\errors
: Everyone full control

DATADIR\trace
: Everyone full control

DATADIR\log

Administrators
Full control

mqm group
Full control

SYSTEM
Full control

Everyone
Read

DATADIR\log\<qmgrname>\active

Administrators
Full control

mqm group
Full control

SYSTEM
Full control

No access granted to Everyone.

Earlier releases of the product

In releases of the product prior to IBM MQ 8.0, the default program and default data directories were co-located.

In any installation that was originally installed before IBM MQ 8.0. and which was installed to the default locations, and then upgraded from that, the data and program directories remain co-located (in `C:\Program Files\IBM\WebSphere MQ`.

In the case of co-located data and program directories, the preceding information applies only to the directories that belong to the data directory, and not those that are part of the program directory.

Naming restrictions for queues

There are restrictions on the length of queue names. Some queue names are reserved for queues defined by the queue manager.

Restrictions on name lengths

Queues can have names up to 48 characters long.

Reserved queue names

Names that start with "SYSTEM." are reserved for queues defined by the queue manager. You can use the **ALTER** or **DEFINE REPLACE** commands to change these queue definitions to suit your installation. The following names are defined for IBM MQ:

Queue Name	Description
SYSTEM.ADMIN.ACTIVITY.QUEUE	Queue for activity reports
SYSTEM.ADMIN.CHANNEL.EVENT	Queue for channel events
SYSTEM.ADMIN.COMMAND.EVENT	Queue for command events
SYSTEM.ADMIN.COMMAND.QUEUE	Queue to which PCF command messages are sent
SYSTEM.ADMIN.CONFIG.EVENT	Queue for configuration events

Queue Name	Description
SYSTEM.ADMIN.PERFM.EVENT	Queue for performance events
SYSTEM.ADMIN.PUBSUB.EVENT	System publish/subscribe related event queue
SYSTEM.ADMIN.QMGR.EVENT	Queue for queue manager events
SYSTEM.ADMIN.TRACE.ROUTE.QUEUE	Queue for trace-route reply messages
SYSTEM.AUTH.DATA.QUEUE	The queue that holds access control lists for the queue manager. (Not for z/OS)
SYSTEM.CHANNEL.INITQ	Initiation queue for channels
SYSTEM.CHANNEL.SYNCQ	The queue that holds the synchronization data for channels
SYSTEM.CHLAUTH.DATA.QUEUE	IBM MQ channel authentication data queue
SYSTEM.CICS.INITIATION.QUEUE	Queue used for triggering (not for z/OS)
SYSTEM.CLUSTER.COMMAND.QUEUE	Queue used to communicate repository changes between queue managers
SYSTEM.CLUSTER.HISTORY.QUEUE	The queue is used to store the history of cluster state information for service purposes.
SYSTEM.CLUSTER.REPOSITORY.QUEUE	Queue used to hold information about the repository
SYSTEM.CLUSTER.TRANSMIT.MODEL.QUEUE	The queue is used to create individual transmit queues for each cluster-sender channel.
SYSTEM.CLUSTER.TRANSMIT.QUEUE	Transmission queue for all destinations managed by cluster support
SYSTEM.COMMAND.INPUT	Queue to which command messages are sent on z/OS
SYSTEM.COMMAND.REPLY.MODEL	Model queue definition for command replies (for z/OS)
SYSTEM.DEAD.LETTER.QUEUE	Dead-letter queue (not for z/OS)
SYSTEM.DEFAULT.ALIAS.QUEUE	Default alias queue definition
SYSTEM.DEFAULT.INITIATION.QUEUE	Queue used to trigger a specified process (not for z/OS)
SYSTEM.DEFAULT.LOCAL.QUEUE	Default local queue definition
SYSTEM.DEFAULT.MODEL.QUEUE	Default model queue definition
SYSTEM.DEFAULT.REMOTE.QUEUE	Default remote queue definition
SYSTEM.DURABLE.SUBSCRIBER.QUEUE	A local queue used to hold a persistent copy of the durable subscriptions in the queue manager
SYSTEM.HIERARCHY.STATE	Queue used to hold information about the state of inter-queue manager relationships in a publish/subscribe hierarchy
SYSTEM.JMS.TEMPQ.MODEL	Model for JMS temporary queues
SYSTEM.INTERNAL.REPLY.QUEUE	IBM MQ internal reply queue (not for z/OS)
SYSTEM.INTER.QMGR.CONTROL	Queue used in a publish/subscribe hierarchy to receive requests from a remote queue manager to create a proxy subscription
SYSTEM.INTER.QMGR.PUBS	Queue used in a publish/subscribe hierarchy to receive publications from a remote queue manager

Queue Name	Description
SYSTEM.INTER.QMGR.FANREQ	Queue used in a publish/subscribe hierarchy to process requests to create a proxy subscription on a remote queue manager
SYSTEM.MQEXPLORER.REPLY.MODEL	Model queue definition for replies for IBM MQ Explorer
SYSTEM.MQSC.REPLY.QUEUE	Model queue definition for MQSC command replies (not for z/OS)
SYSTEM.QSG.CHANNEL.SYNCQ	Shared local queue used for storing messages that contain the synchronization information for shared channels (z/OS only)
SYSTEM.QSG.TRANSMIT.QUEUE	Shared local queue used by the intra-group queuing agent when transmitting messages between queue managers in the same queue sharing group (z/OS only)
SYSTEM.RETAINED.PUB.QUEUE	A local queue used to hold a copy of each retained publication in the queue manager.
SYSTEM.SELECTION.EVALUATION.QUEUE	IBM MQ internal selection evaluation queue (not for z/OS)
SYSTEM.SELECTION.VALIDATION.QUEUE	IBM MQ internal selection validation queue (not for z/OS)

Naming restrictions for other objects

There are restrictions on the length of object names. Some object names are reserved for objects defined by the queue manager.

Restrictions on name length

Processes, namelists, clusters, topics, services, and authentication information objects can have names up to 48 characters long.

Channels can have names up to 20 characters long.

Storage classes can have names up to 8 characters long.

CF structures can have names up to 12 characters long.

Reserved object names

Names that start with SYSTEM. are reserved for objects defined by the queue manager. You can use the **ALTER** or **DEFINE REPLACE** commands to change these object definitions to suit your installation. The following names are defined for IBM MQ:

Object Name	Description
SYSTEM.ADMIN.SVRCONN	Server-connection channel used for remote administration of a queue manager
SYSTEM.AUTO.RECEIVER	Default receiver channel for auto definition (UNIX, Linux, and Windows systems only)
SYSTEM.AUTO.SVRCONN	Default server-connection channel for auto definition (Multiplatforms only)

Object Name	Description
SYSTEM.BASE.TOPIC	Base topic for ASPARENT resolution. If a particular administrative topic object has no parent administrative topic objects, any ASPARENT attributes are inherited from this object
SYSTEM.DEF.CLNTCONN	Default client-connection channel definition
SYSTEM.DEF.CLUSRCVR	Default cluster-receiver channel definition
SYSTEM.DEF.CLUSSDR	Default cluster-sender channel definition
SYSTEM.DEF.RECEIVER	Default receiver channel definition
SYSTEM.DEF.REQUESTER	Default requester channel definition
SYSTEM.DEF.SENDER	Default sender channel definition
SYSTEM.DEF.SERVER	Default server channel definition
SYSTEM.DEF.SVRCONN	Default server-connection channel definition
SYSTEM.DEFAULT.AUTHINFO.CRLLDAP	Default authentication information object definition for defining authentication information objects of type CRLLDAP
SYSTEM.DEFAULT.AUTHINFO.OCSP	Default authentication information object definition for defining authentication information objects of type OCSP
SYSTEM.DEFAULT.LISTENER.LU62	Default SNA listener (Windows only)
SYSTEM.DEFAULT.LISTENER.NETBIOS	Default NetBIOS listener (Windows only)
SYSTEM.DEFAULT.LISTENER.SPX	Default SPX listener (Windows only)
SYSTEM.DEFAULT.LISTENER.TCP	Default TCP/IP listener (Multiplatforms only)
SYSTEM.DEFAULT.NAMELIST	Default namelist definition
SYSTEM.DEFAULT.PROCESS	Default process definition
SYSTEM.DEFAULT.SEVICE	Default service (Multiplatforms only)
SYSTEM.DEFAULT.TOPIC	Default topic definition
SYSTEM.QPUBSUB.QUEUE.NAMELIST	A list of queues for the Queued Publish/Subscribe interface to monitor
z/OS z/OS SYSTEMST	Default storage class definition (z/OS only)

Queue name resolution

This topic contains information about queue name resolution as performed by queue managers at both sending and receiving ends of a channel.

In larger networks, the use of queue managers has a number of advantages over other forms of communication. These advantages derive from the name resolution function in DQM and the main benefits are:

- Applications do not need to make routing decisions
- Applications do not need to know the network structure
- Network links are created by systems administrators
- Network structure is controlled by network planners
- Multiple channels can be used between nodes to partition traffic

The following figure shows an example of queue name resolution. The figure shows two machines in a network, one running a put application, the other running a get application. The applications communicate with each other through the IBM MQ channel, controlled by the MCAs.

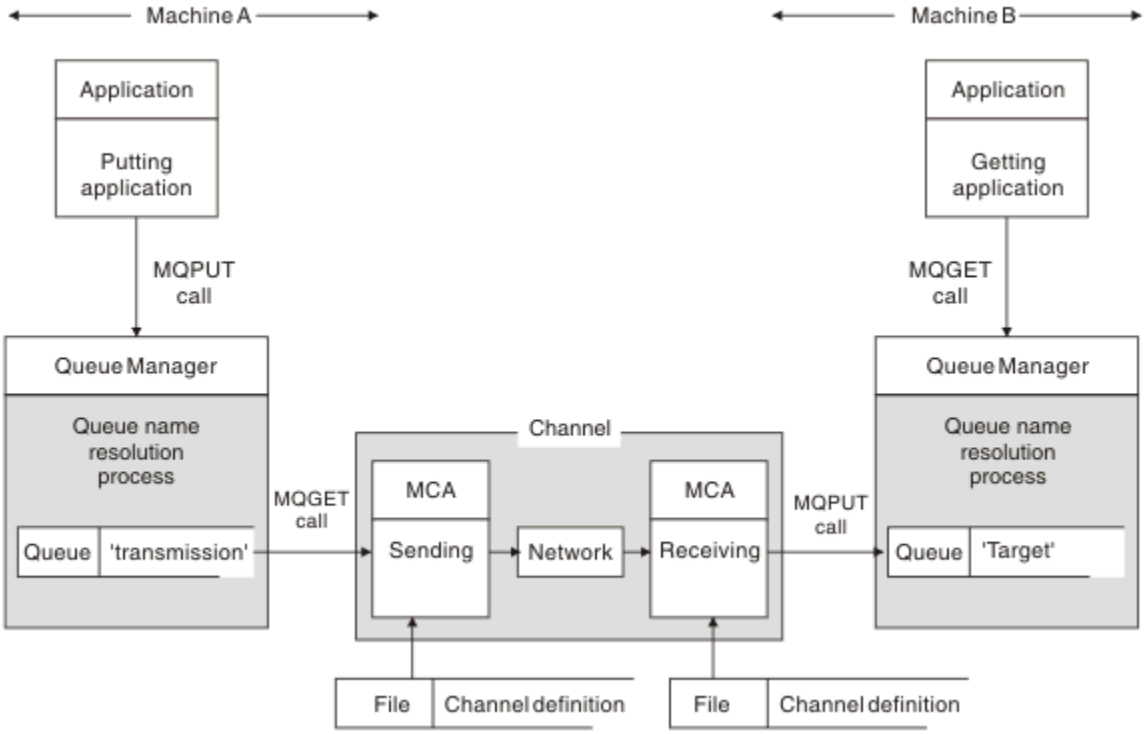

Figure 5. Name resolution

Referring to Figure 5 on page 86, the basic mechanism for putting messages on a remote queue, as far as the application is concerned, is the same as for putting messages on a local queue:

- The application putting the message issues MQOPEN and MQPUT calls to put messages on the target queue.
- The application getting the messages issues MQOPEN and MQGET calls to get the messages from the target queue.

If both applications are connected to the same queue manager then no inter-queue manager communication is required, and the target queue is described as *local* to both applications.

However, if the applications are connected to different queue managers, two MCAs and their associated network connection are involved in the transfer, as shown in the figure. In this case, the target queue is considered to be a *remote queue* to the putting application.

The sequence of events is as follows:

1. The putting application issues MQOPEN and MQPUT calls to put messages to the target queue.
2. During the MQOPEN call, the *name resolution* function detects that the target queue is not local, and decides which transmission queue is appropriate. Thereafter, on the MQPUT calls associated with the MQOPEN call, all messages are placed on this transmission queue.
3. The sending MCA gets the messages from the transmission queue and passes them to the receiving MCA at the remote computer.
4. The receiving MCA puts the messages on the target queue, or queues.
5. The getting application issues MQOPEN and MQGET calls to get the messages from the target queue.

Note: Only step 1 and step 5 involve application code; steps 2 through 4 are performed by the local queue managers and the MCA programs. The putting application is unaware of the location of the target queue, which could be in the same processor, or in another processor on another continent.

The combination of sending MCA, the network connection, and the receiving MCA, is called a *message channel*, and is inherently a unidirectional device. Normally, it is necessary to move messages in both directions, and two channels are set up for this movement, one in each direction.

What is queue name resolution?

Queue name resolution is vital to DQM. It removes the need for applications to be concerned with the physical location of queues, and insulates them against the details of networks.

A systems administrator can move queues from one queue manager to another, and change the routing between queue managers without applications needing to know anything about it.

In order to uncouple from the application design the exact path over which the data travels, it is necessary to introduce a level of indirection between the name used by the application when it refers to the target queue, and the naming of the channel over which the flow occurs. This indirection is achieved using the queue name resolution mechanism.

In essence, when an application refers to a queue name, the name is mapped by the resolution mechanism either to a transmission queue or to a local queue that is not a transmission queue. For mapping to a transmission queue, a second name resolution is needed at the destination, and the received message is placed on the target queue as intended by the application designer. The application remains unaware of the transmission queue and channel used for moving the message.

Note: The definition of the queue and channel is a system management responsibility and can be changed by an operator or a system management utility, without the need to change applications.

An important requirement for the system management of message flows is that alternative paths need to be provided between queue managers. For example, business requirements might dictate that different *classes of service* are sent over different channels to the same destination. This decision is a system management decision and the queue name resolution mechanism provides a flexible way to achieve it. The Application Programming Guide describes this in detail, but the basic idea is to use queue name resolution at the sending queue manager to map the queue name supplied by the application to the appropriate transmission queue for the type of traffic involved. Similarly at the receiving end, queue name resolution maps the name in the message descriptor to a local (not a transmission) queue or again to an appropriate transmission queue.

Not only is it possible for the forward path from one queue manager to another to be partitioned into different types of traffic, but the return message that is sent to the reply-to queue definition in the outbound message can also use the same traffic partitioning. Queue name resolution satisfies this requirement and the application designer need not be involved in these traffic partitioning decisions.

The point that the mapping is carried out at both the sending and receiving queue managers is an important aspect of the way name resolution works. This mapping allows the queue name supplied by the putting application to be mapped to a local queue or a transmission queue at the sending queue manager, and again remapped to a local queue or a transmission queue at the receiving queue manager.

Reply messages from receiving applications or MCAs have the name resolution carried out in the same way, allowing return routing over specific paths with queue definitions at all the queue managers on route.

System and default objects

Lists the system and default objects created by the **crtmqm** command.

When you create a queue manager using the **crtmqm** control command, the system objects and the default objects are created automatically.

- The system objects are those IBM MQ objects needed to operate a queue manager or channel.
- The default objects define all the attributes of an object. When you create an object, such as a local queue, any attributes that you do not specify explicitly are inherited from the default object.

The following tables list the system and default objects created by **crtmqm**:

- Table 11 on page 88 lists the system and default queue objects.

- Table 12 on page 89 lists the system and default topic objects.
- Table 13 on page 89 lists the system and default channel objects.
- Table 14 on page 90 lists the system and default authentication information objects.
- Table 15 on page 90 lists the system and default listener objects.
- Table 16 on page 90 lists the system and default namelist objects.
- Table 17 on page 90 lists the system and default process objects.
- Table 18 on page 90 lists the system and default service objects.

Table 11. System and default objects: queues

Object name	Description
SYSTEM.ADMIN.ACCOUNTING.QUEUE	The queue that holds accounting monitoring data.
SYSTEM.ADMIN.ACTIVITY.QUEUE	The queue that holds returned activity reports.
SYSTEM.ADMIN.CHANNEL.EVENT	Event queue for channels.
SYSTEM.ADMIN.COMMAND.EVENT	Event queue for command events.
SYSTEM.ADMIN.COMMAND.QUEUE	Administration command queue. Used for remote MQSC commands and PCF commands.
SYSTEM.ADMIN.CONFIG.EVENT	Event queue for configuration events.
SYSTEM.ADMIN.PERFM.EVENT	Event queue for performance events.
SYSTEM.ADMIN.PUBSUB.EVENT	System publish/subscribe related event queue
SYSTEM.ADMIN.QMGR.EVENT	Event queue for queue manager events.
SYSTEM.ADMIN.STATISTICS.QUEUE	The queue that holds statistics monitoring data.
SYSTEM.ADMIN.TRACE.ACTIVITY.QUEUE	The queue that displays trace activity.
SYSTEM.ADMIN.TRACE.ROUTE.QUEUE	The queue that holds returned trace-route reply messages.
SYSTEM.AUTH.DATA.QUEUE	The queue that holds access control lists for the queue manager.
SYSTEM.CHANNEL.INITQ	Channel initiation queue.
SYSTEM.CHANNEL.SYNCQ	The queue that holds the synchronization data for channels.
SYSTEM.CHLAUTH.DATA.QUEUE	IBM MQ channel authentication data queue
SYSTEM.CICS.INITIATION.QUEUE	Default CICS initiation queue.
SYSTEM.CLUSTER.COMMAND.QUEUE	The queue used to carry messages to the repository queue manager.
SYSTEM.CLUSTER.HISTORY.QUEUE	The queue is used to store the history of cluster state information for service purposes.
SYSTEM.CLUSTER.TRANSMIT.MODEL.QUEUE	The queue is used to create individual transmit queues for each cluster-sender channel.
SYSTEM.CLUSTER.REPOSITORY.QUEUE	The queue used to store all repository information.
SYSTEM.CLUSTER.TRANSMIT.QUEUE	The transmission queue for all messages to all clusters.
SYSTEM.DEAD.LETTER.QUEUE	Dead-letter (undelivered-message) queue.

Table 11. System and default objects: queues (continued)	
Object name	**Description**
SYSTEM.DEFAULT.ALIAS.QUEUE	Default alias queue.
SYSTEM.DEFAULT.INITIATION.QUEUE	Default initiation queue.
SYSTEM.DEFAULT.LOCAL.QUEUE	Default local queue.
SYSTEM.DEFAULT.MODEL.QUEUE	Default model queue.
SYSTEM.DEFAULT.REMOTE.QUEUE	Default remote queue.
SYSTEM.JMS.TEMPQ.MODEL	Model for JMS temporary queues
SYSTEM.MQEXPLORER.REPLY.MODEL	The IBM MQ Explorer reply-to queue. This is a model queue that creates a temporary dynamic queue for replies to the IBM MQ Explorer.
SYSTEM.MQSC.REPLY.QUEUE	MQSC command reply-to queue. This is a model queue that creates a temporary dynamic queue for replies to remote MQSC commands.
SYSTEM.PENDING.DATA.QUEUE	Support deferred messages in JMS.

Table 12. System and default objects: topics	
Object name	**Description**
SYSTEM.BASE.TOPIC	Base topic for ASPARENT resolution. If a particular topic has no parent administrative topic objects, or those parent objects also have ASPARENT, any remaining ASPARENT attributes are inherited from this object.
SYSTEM.DEFAULT.TOPIC	Default topic definition.

Table 13. System and default objects: channels	
Object name	**Description**
SYSTEM.AUTO.RECEIVER	Dynamic receiver channel.
SYSTEM.AUTO.SVRCONN	Dynamic server-connection channel.
SYSTEM.DEF.CLUSRCVR	Default receiver channel for the cluster, used to supply default values for any attributes not specified when a CLUSRCVR channel is created on a queue manager in the cluster.
SYSTEM.DEF.CLUSSDR	Default sender channel for the cluster, used to supply default values for any attributes not specified when a CLUSSDR channel is created on a queue manager in the cluster.
SYSTEM.DEF.RECEIVER	Default receiver channel.
SYSTEM.DEF.REQUESTER	Default requester channel.
SYSTEM.DEF.SENDER	Default sender channel.
SYSTEM.DEF.SERVER	Default server channel.
SYSTEM.DEF.SVRCONN	Default server-connection channel.

Table 13. System and default objects: channels (continued)	
Object name	**Description**
SYSTEM.DEF.CLNTCONN	Default client-connection channel.
V 9.0.0 V 9.0.0 SYSTEM.DEF.AMQP	Default AMQP channel.

Table 14. System and default objects: authentication information objects	
Object name	**Description**
SYSTEM.DEFAULT.AUTHINFO.CRLLDAP	Default authentication information object for defining authentication information objects of type CRLLDAP.
SYSTEM.DEFAULT.AUTHINFO.OCSP	Default authentication information object for defining authentication information objects of type OCSP.

Table 15. System and default objects: listeners	
Object name	**Description**
SYSTEM.DEFAULT.LISTENER.TCP	Default TCP listener.
SYSTEM.DEFAULT.LISTENER.LU62 [1]	Default LU62 listener.
SYSTEM.DEFAULT.LISTENER.NETBIOS [1]	Default NETBIOS listener.
SYSTEM.DEFAULT.LISTENER.SPX [1]	Default SPX listener.

1. Windows only

Table 16. System and default objects: namelists	
Object name	**Description**
SYSTEM.DEFAULT.NAMELIST	Default namelist.

Table 17. System and default objects: processes	
Object name	**Description**
SYSTEM.DEFAULT.PROCESS	Default process definition.

Table 18. System and default objects: services	
Object name	**Description**
SYSTEM.DEFAULT.SERVICE	Default service.
SYSTEM.BROKER	Publish/subscribe broker

Windows Windows default configuration objects

On Windows systems, you can set up a default configuration using the IBM MQ Postcard application.

Note: You cannot set up a default configuration if other queue managers exist on your computer.

Many of the names used for the Windows default configuration objects involve the use of a short TCP/IP name. This is the TCP/IP name of the computer, without the domain part; for example the short TCP/IP name for the computer `mycomputer.hursley.ibm.com` is `mycomputer`. In all cases, where this name has to be truncated, if the last character is a period (.), it is removed.

Any characters within the short TCP/IP name that are not valid for IBM MQ object names (for example, hyphens) are replaced by an underscore character.

Valid characters for IBM MQ object names are: a to z, A to Z, 0 to 9, and the four special characters / % . and _.

The cluster name for the Windows default configuration is DEFAULT_CLUSTER.

If the queue manager is not a repository queue manager, the objects listed in Table 19 on page 91 are created.

Table 19. Objects created by the Windows default configuration application

Object	**Name**
Queue manager	The short TCP/IP name prefixed with the characters QM_. The maximum length of the queue manager name is 48 characters. Names exceeding this limit are truncated at 48 characters. If the last character of the name is a period (.), this is replaced by a space ( ). The queue manager has a command server, a channel listener, and channel initiator associated with it. The channel listener listens on the standard IBM MQ port, port number 1414. Any other queue managers created on this machine must not use port 1414 while the default configuration queue manager still exists.
Generic cluster receiver channel	The short TCP/IP name prefixed with the characters TO_QM_. The maximum length of the generic cluster receiver name is 20 characters. Names exceeding this limit are truncated at 20 characters. If the last character of the name is a period (.), this is replaced by a space ( ).
Cluster sender channel	The cluster sender channel is initially created with the name TO_+QMNAME+. When IBM MQ has established a connection to the repository queue manager for the default configuration cluster, this name is replaced with the name of the repository queue manager for the default configuration cluster, prefixed with the characters TO_. The maximum length of the cluster sender channel name is 20 characters. Names exceeding this limit are truncated at 20 characters. If the last character of the name is a period (.), this is replaced by a space ( ).
Local message queue	The local message queue is called `default`.
Local message queue for use by the IBM MQ Postcard application	The local message queue for use by the IBM MQ Postcard application is called `postcard`.
Server connection channel	The server connection channel allows clients to connect to the queue manager. Its name is the short TCP/IP name, prefixed with the characters S_. The maximum length of the server connection channel name is 20 characters. Names exceeding this limit are truncated at 20 characters. If the last character of the name is a period (.), this is replaced by a space ( ).

If the queue manager is a repository queue manager, the default configuration is similar to that described in Table 19 on page 91, but with the following differences:

- The queue manager is defined as a repository queue manager for the default configuration cluster.
- There is no cluster-sender channel defined.

- A local cluster queue that is the short TCP/IP name prefixed with the characters clq_default_ is created. The maximum length of this name is 48 characters. Names exceeding this length are truncated at 48 characters.

If you request remote administration facilities, the server connection channel, SYSTEM.ADMIN.SVRCONN is also created.

SYSTEM.BASE.TOPIC

Base topic for ASPARENT resolution. If a particular topic has no parent administrative topic objects, or those parent objects also have ASPARENT, any remaining ASPARENT attributes are inherited from this object.

Table 20. Default values of SYSTEM.BASE.TOPIC

Parameter	Value
TOPICSTR	"
CLROUTE	DIRECT
CLUSTER	The default value is an empty string.
COMMINFO	SYSTEM.DEFAULT.COMMINFO.MULTICAST
DEFPRESP	SYNC
DEFPRTY	0
DEFPSIST	NO
DESCR	'Base topic for resolving attributes'
DURSUB	YES
MCAST	DISABLED
MDURMDL	SYSTEM.DURABLE.MODEL.QUEUE
MNDURMDL	SYSTEM.NDURABLE.MODEL.QUEUE
NPMSGDLV	ALLAVAIL
PMSGDLV	ALLDUR
PROXYSUB	FIRSTUSE
PUB	ENABLED
PUBSCOPE	ALL
z/OS z/OS QSGDISP (z/OS platform only)	QMGR
SUB	ENABLED
SUBSCOPE	ALL
USEDLQ	YES
WILDCARD	PASSTHRU

If this object does not exist, its default values are still used by IBM MQ for ASPARENT attributes that are not resolved by parent topics further up the topic tree.

Setting the PUB or SUB attributes of SYSTEM.BASE.TOPIC to DISABLED prevents applications publishing or subscribing to topics in the topic tree, with two exceptions:

1. Any topic objects in the topic tree that have PUB or SUB explicitly set to ENABLE. Applications can publish or subscribe to those topics, and their children.
2. Publication and subscription to SYSTEM.BROKER.ADMIN.STREAM is not disabled by the setting the PUB or SUB attributes of SYSTEM.BASE.TOPIC to DISABLED.

See also Special handling for the **PUB** parameter.

IBM i System and default objects for IBM i

When you create a queue manager using the CRTMQM command, the system objects and the default objects are created automatically.

- The system objects are those IBM MQ objects required for the operation of a queue manager or channel.
- The default objects define all the attributes of an object. When you create an object, such as a local queue, any attributes that you do not specify explicitly are inherited from the default object.

The following tables list the system and default objects created by **CRTMQM**:

- Table 21 on page 93 lists the system and default queue objects.
- Table 22 on page 95 lists the system and default channel objects.
- Table 23 on page 95 gives the system and default authentication information objects.
- Table 24 on page 95 gives the system and default listener object.
- Table 25 on page 96 gives the system and default namelist object.
- Table 26 on page 96 gives the system and default process object.
- Table 27 on page 96 gives the system and default service object.

Table 21. System and default objects: queues

Object name	Description
SYSTEM.ADMIN.ACCOUNTING.QUEUE	Accounting message data generated when an application disconnects from the queue manager.
SYSTEM.ADMIN.ACTIVITY.QUEUE	Activity report message data.
SYSTEM.ADMIN.CHANNEL.EVENT	Event queue for channels.
SYSTEM.ADMIN.COMMAND.QUEUE	Administration command queue. Used for remote MQSC commands and PCF commands.
SYSTEM.ADMIN.LOGGER.EVENT	Logger event (journal receiver) message data.
SYSTEM.ADMIN.PERFM.EVENT	Event queue for performance events.
SYSTEM.ADMIN.PUBSUB.EVENT	System publish/subscribe related event queue
SYSTEM.ADMIN.QMGR.EVENT	Event queue for queue manager events.
SYSTEM.ADMIN.STATISTICS.QUEUE	MQI, queue and channel statistics message data queue.
SYSTEM.ADMIN.TRACE.ROUTE.QUEUE	Trace-route reply message data queue.
SYSTEM.AUTH.DATA.QUEUE	Used by the object authority manager (OAM).
SYSTEM.BROKER.ADMIN.STREAM	Admin stream used by the queued publish/subscribe interface.
SYSTEM.BROKER.CONTROL.QUEUE	Publish/subscribe interface control queue.
SYSTEM.BROKER.DEFAULT.STREAM	The default stream used by the queued publish/subscribe interface.

Table 21. System and default objects: queues (continued)	
Object name	**Description**
SYSTEM.BROKER.INTER.BROKER.COMMUNICATIONS	Broker to broker communications queue.
SYSTEM.CHANNEL.INITQ	Channel initiation queue.
SYSTEM.CHANNEL.SYNCQ	The queue that holds the synchronization data for channels.
SYSTEM.CHLAUTH.DATA.QUEUE	IBM MQ channel authentication data queue
SYSTEM.DURABLE.MODEL.QUEUE	A queue used as a model for managed durable subscriptions.
SYSTEM.DURABLE.SUBSCRIBER.QUEUE	A queue used to hold a persistent copy of the durable subscriptions in the queue manager.
SYSTEM.CICS.INITIATION.QUEUE	Default CICS initiation queue.
SYSTEM.CLUSTER.COMMAND.QUEUE	The queue used to carry messages to the repository queue manager.
SYSTEM.CLUSTER.HISTORY.QUEUE	The queue is used to store the history of cluster state information for service purposes.
SYSTEM.CLUSTER.REPOSITORY.QUEUE	The queue used to store all repository information.
SYSTEM.CLUSTER.TRANSMIT.MODEL.QUEUE	The queue is used to create individual transmit queues for each cluster-sender channel.
SYSTEM.CLUSTER.TRANSMIT.QUEUE	The transmission queue for all messages to all clusters.
SYSTEM.DEAD.LETTER.QUEUE	Dead-letter (undelivered message) queue.
SYSTEM.DEFAULT.ALIAS.QUEUE	Default alias queue.
SYSTEM.DEFAULT.AUTHINFO.CRLLDAP	Default authentication information definition.
SYSTEM.DEFAULT.INITIATION.QUEUE	Default initiation queue.
SYSTEM.DEFAULT.LOCAL.QUEUE	Default local queue.
SYSTEM.DEFAULT.MODEL.QUEUE	Default model queue.
SYSTEM.DEFAULT.REMOTE.QUEUE	Default remote queue.
SYSTEM.JMS.TEMPQ.MODEL	Model for JMS temporary queues
SYSTEM.HIERARCHY.STATE	IBM MQ distributed publish/subscribe hierarchy relationship state.
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.MQEXPLORER.REPLY.MODEL	IBM MQ Explorer reply-to queue. This is a model queue that creates a temporary dynamic queue for replies to the IBM MQ Explorer.

Table 21. System and default objects: queues (continued)	
Object name	**Description**
SYSTEM.MQSC.REPLY.QUEUE	MQSC command reply-to queue. This is a model queue that creates a temporary dynamic queue for replies to remote MQSC commands.
SYSTEM.NDURABLE.MODEL.QUEUE	A queue used as a model for managed non durable subscriptions.
SYSTEM.PENDING.DATA.QUEUE	Support deferred messages in JMS.
SYSTEM.RETAINED.PUB.QUEUE	A queue used to hold a copy of each retained publication in the queue manager.

Table 22. System and default objects: channels	
Object name	**Description**
SYSTEM.AUTO.RECEIVER	Dynamic receiver channel.
SYSTEM.AUTO.SVRCONN	Dynamic server-connection channel.
SYSTEM.DEF.CLNTCONN	Default client connection channel, used to supply default values for any attributes not specified when a CLNTCONN channel is created on a queue manager.
SYSTEM.DEF.CLUSRCVR	Default receiver channel for the cluster used to supply default values for any attributes not specified when a CLUSRCVR channel is created on a queue manager in the cluster.
SYSTEM.DEF.CLUSSDR	Default sender channel for the cluster used to supply default values for any attributes not specified when a CLUSSDR channel is created on a queue manager in the cluster.
SYSTEM.DEF.RECEIVER	Default receiver channel.
SYSTEM.DEF.REQUESTER	Default requester channel.
SYSTEM.DEF.SENDER	Default sender channel.
SYSTEM.DEF.SERVER	Default server channel.
SYSTEM.DEF.SVRCONN	Default server-connection channel.

Table 23. System and default objects: authentication information objects	
Object name	**Description**
SYSTEM.DEFAULT.AUTHINFO.CRLLDAP	Default authentication information object for authentication type CRLLDAP.
SYSTEM.DEFAULT.AUTHINFO.OCSP	Default authentication information object for authentication type OCSP.

Table 24. System and default objects: listeners	
Object name	**Description**
SYSTEM.DEFAULT.LISTENER.TCP	Default listener for TCP transport.

Table 25. System and default objects: namelists	
Object name	**Description**
`SYSTEM.DEFAULT.NAMELIST`	Default namelist definition.
`SYSTEM.QPUBSUB.QUEUE.NAMELIST`	A list of queue names monitored by the queued publish/subscribe interface.
`SYSTEM.QPUBSUB.SUBPOINT.NAMELIST`	A list of topic objects used by the queued publish/subscribe interface to match topic objects to subscription points.

Table 26. System and default objects: processes	
Object name	**Description**
`SYSTEM.DEFAULT.PROCESS`	Default process definition.

Table 27. System and default objects: services	
Object name	**Description**
`SYSTEM.DEFAULT.SERVICE`	Default service.

Stanza information

The following information helps you configure the information within stanzas, and lists the contents of the `mqs.ini`, `qm.ini`, and `mqclient.ini` files.

Configuring stanzas

Use the links to help you configure the system, or systems, in your enterprise:

- Changing IBM MQ configuration information helps you configure the:
  - *AllQueueManagers* stanza
  - *DefaultQueueManager* stanza
  - *ExitProperties* stanza
  - *LogDefaults* stanza
  - *Security* stanza in the `qm.ini` file
- Changing queue manager configuration information helps you configure the:
  - Windows *AccessMode* stanza (Windows only)
  - *Service* stanza - for Installable services
  - *Log* stanza
  - Linux UNIX *RestrictedMode* stanza (UNIX and Linux systems only)
  - *XAResourceManager* stanza
  - *TCP*, *LU62*, and *NETBIOS* stanzas
  - *ExitPath* stanza
  - *QMErrorLog* stanza
  - *SSL* stanza
  - *ExitPropertiesLocal* stanza
- Configuring services and components helps you configure the:
  - *Service* stanza

– *ServiceComponent* stanza

and contains links to how they are used for different services on UNIX and Linux, and Windows platforms.

- Configuring API exits helps you configure the:
  – *AllActivityTrace* stanza
  – *AppplicationTrace* stanza
- Configuring activity trace behavior helps you configure the:
  – *ApiExitCommon* stanza
  – *ApiExitTemplate* stanza
  – *ApiExitLocal* stanza
- Configuration information for clients helps you configure the:
  – *CHANNELS* stanza
  – *ClientExitPath* stanza
  – Windows *LU62*, *NETBIOS* and *SPX* stanza (Windows only)
  – *MessageBuffer* stanza
  – *SSL* stanza
  – *TCP* stanza
- "Configuration file stanzas for distributed queuing" on page 98 helps you configure the:
  – *CHANNELS* stanza
  – *TCP* stanza
  – *LU62* stanza
  – *NETBIOS*
  – *ExitPath* stanza
- Setting queued publish/subscribe message attributes helps you configure the:
  – *PersistentPublishRetry* attribute
  – *NonPersistentPublishRetry* attribute
  – *PublishBatchSize* attribute
  – *PublishRetryInterval* attribute

  in the *Broker* stanza.

Attention: You must create a *Broker* stanza if you need one.

Configuration files

See:

- **mqs.ini** file
- **qm.ini** file
- **mqclient.ini** file

for a list of the possible stanzas in each configuration file. Linux UNIX

mqs.ini file

Example of an IBM MQ configuration file for UNIX and Linux systems shows an example `mqs.ini` file.

An `mqs.ini` file can contain the following stanzas:

- *AllQueueManagers*

- *DefaultQueueManager*
- *ExitProperties*
- *LogDefaults*

In addition, there is one *QueueManager* stanza for each queue manager.

qm.ini file

Example queue manager configuration file for IBM MQ for UNIX and Linux systems shows an example `qm.ini` file.

A `qm.ini` file can contain the following stanzas:

- *ExitPath*
- *Log*
- *QMErrorLog*
- *QueueManager*
- *Security*
- *ServiceComponent*

  Multi To configure *InstallableServices* use the *Service* and *ServiceComponent* stanzas.
- *Connection* for *DefaultBindType*

  Attention: You must create a *Connection* stanza if you need one.
- *SSL and TLS*
- *TCP, LU62, and NETBIOS*
- XAResourceManager

In addition, you can change the:

- Windows *AccessMode* (Windows only)
- Linux UNIX *RestrictedMode* (UNIX and Linux systems only)

by using the crtmqm command.

mqclient.ini file

An `mqclient.ini` file can contain the following stanzas:

- *CHANNELS*
- *ClientExitPath*
- *LU62*, *NETBIOS*, and *SPX*
- *MessageBuffer*
- *SSL*
- *TCP*

In addition, you might need a PreConnect stanza to configure a preconnect exit.

Configuration file stanzas for distributed queuing

A description of the stanzas of the queue manager configuration file, qm.ini, related to distributed queuing.

This topic shows the stanzas in the queue manager configuration file that relate to distributed queuing. It applies to the queue manager configuration file for IBM MQ for Multiplatforms. The file is called `qm.ini` on all platforms.

The stanzas that relate to distributed queuing are:

- CHANNELS
- TCP
- LU62
- NETBIOS
- EXITPATH

Figure 6 on page 99 shows the values that you can set using these stanzas. When you are defining one of these stanzas, 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.

```
CHANNELS:
 MAXCHANNELS=n            ; Maximum number of channels allowed, the
                          ; default value is 100.
 MAXACTIVECHANNELS=n      ; Maximum number of channels allowed to be active at
                          ; any time, the default is the value of MaxChannels.
 MAXINITIATORS=n          ; Maximum number of initiators allowed, the default
                          ; and maximum value is 3.
 MQIBINDTYPE=type         ; Whether the binding for applications is to be
                          ; "fastpath" or "standard".
                          ; The default is "standard".
 PIPELINELENGTH=n         ; The maximum number of concurrent threads a channel will use.
                          ; The default is 1. Any value greater than 1 is treated as 2.
 ADOPTNEWMCA=chltype      ; Stops previous process if channel fails to start.
                          ; The default is "NO".
 ADOPTNEWMCATIMEOUT=n     ; Specifies the amount of time that the new
                          ; process should wait for the old process to end.
                          ; The default is 60.
 ADOPTNEWMCACHECK=        ; Specifies the type checking required.
           typecheck      ; The default is "NAME","ADDRESS", and "QM".
 CHLAUTHEARLYADOPT=Y/N    ; The order in which connection authentication and channel
authentication rules are
                          ; processed. If not present in the qm.ini file the default is "N".
From MQ9.0.4 all
                          ; queue managers are created with a defa\ult of "Y"
 PASSWORDPROTECTION=      ; From MQ8.0,set protected passwords in the MQCSP structure, rather
than using TLS.
             options      ; The options are "compatible", "always", "optional" and "warn"
                          ; The default is "compatible".
 CHLAUTHISSUEWARN=Y       ; If you want message AMQ9787 to be generated when you set theWARN=YES
attribute
                          ; on the SET CHLAUTH command.
TCP:                      ; TCP entries
 PORT=n                   ; Port number, the default is 1414
 KEEPALIVE=Yes            ; Switch TCP/IP KeepAlive on
LU62:
 LIBRARY2=DLLName2        ; Used if code is in two libraries
EXITPATH:[1]     Location of user exits
 EXITPATHS=             ; String of directory paths.
```

Figure 6. qm.ini stanzas for distributed queuing

Notes:

1. EXITPATH applies only to the following platforms:
   - AIX
   - HP-UX
   - Solaris
   - Windows

Related tasks

Configuring

z/OS Configuring z/OS

Changing configuration information on Windows, UNIX, and Linux systems

IBM i Changing configuration information on IBM i

Channel attributes

This section describes the channel attributes held in the channel definitions.

You choose the attributes of a channel to be optimal for a particular set of circumstances for each channel. However, when the channel is running, the actual values might have changed during startup negotiations. See Preparing channels.

Many attributes have default values, and you can use these values for most channels. However, in those circumstances where the defaults are not optimal, see this section for guidance in selecting the correct values.

For cluster channels, you specify the cluster channel attributes on the cluster-receiver channels at the target queue managers. Any attributes you specify on the matching cluster-sender channels are likely to be ignored. See Cluster channels.

Note: In IBM MQ for IBM i, most attributes can be specified as `*SYSDFTCHL`, which means that the value is taken from the system default channel in your system.

Channel attributes and channel types

Different types of channel support different channel attributes.

The channel types for IBM MQ channel attributes are listed in the following table.

Note: For cluster channels (the CLUSSDR and CLUSRCVR columns in the table), if an attribute can be set on both channels, set it on both and ensure that the settings are identical. If there is any discrepancy between the settings, those that you specify on the CLUSRCVR channel are likely to be used. This is explained in Cluster channels.

Table 28. Channel attributes for the channel types

Attribute field	MQSC command parameter	SDR	SVR	RCVR	RQSTR	CLNT-CONN	SVR-CONN	CLUS-SDR	CLUS-RCVR	V 9.0.0 AMQP
Alter date	ALTDATE	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	V 9.0.0 Yes
Alter time	ALTTIME	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	V 9.0.0 Yes
V 9.0.0 Multi V 9.0.0 Multi AMQP keep alive	AMQPKA									V 9.0.0 Yes
Batch heartbeat interval	BATCHHB	Yes	Yes					Yes	Yes	
Batch interval	BATCHINT	Yes	Yes					Yes	Yes	
Batch limit	BATCHLIM	Yes	Yes					Yes	Yes	
Batch size	BATCHSZ	Yes	Yes	Yes	Yes			Yes	Yes	
Certificate label	CERTLABL	Yes	Yes	Yes	Yes	Yes	Yes	Yes "1" on page 104	Yes	V 9.0.0 Yes

Table 28. Channel attributes for the channel types (continued)										
Attribute field	**MQSC command parameter**	**SDR**	**SVR**	**RCVR**	**RQSTR**	**CLNT-CONN**	**SVR-CONN**	**CLUS-SDR**	**CLUS-RCVR**	**V 9.0.0 AMQP**
Channel name	CHANNEL	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	V 9.0.0 Yes
Channel statistics	STATCHL	Yes	Yes	Yes	Yes			Yes	Yes	
Channel type	CHLTYPE	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	V 9.0.0 Yes
Client channel weight	CLNTWGHT					Yes				
Cluster	CLUSTER							Yes	Yes	
Cluster namelist	CLUSNL							Yes	Yes	
Cluster workload priority	CLWLPRTY							Yes	Yes	
Cluster workload rank	CLWLRANK							Yes	Yes	
Cluster workload weight	CLWLWGHT							Yes	Yes	
Connection affinity	AFFINITY					Yes				
Connection name	CONNAME	Yes	Yes		Yes	Yes		Yes	Yes	
Convert message	CONVERT	Yes	Yes					Yes	Yes	
Data compression	COMPMSG	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	
Description	DESCR	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	V 9.0.0 Yes
Disconnect interval	DISCINT	Yes	Yes				Yes "2" on page 104	Yes	Yes	
Disposition "2" on page 104	QSGDISP	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	
Header compression	COMPHDR	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	
Heartbeat interval	HBINT	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	
Keepalive interval	KAINT	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	
Local address	LOCLADDR	Yes	Yes		Yes	Yes		Yes	Yes	V 9.0.0 Yes
Long retry count	LONGRTY	Yes	Yes					Yes	Yes	
Long retry interval	LONGTMR	Yes	Yes					Yes	Yes	
LU 6.2 mode name	MODENAME	Yes	Yes		Yes	Yes		Yes	Yes	

Table 28. Channel attributes for the channel types (continued)										
Attribute field	**MQSC command parameter**	**SDR**	**SVR**	**RCVR**	**RQSTR**	**CLNT-CONN**	**SVR-CONN**	**CLUS-SDR**	**CLUS-RCVR**	**V 9.0.0 AMQP**
LU 6.2 transaction program name	TPNAME	Yes	Yes		Yes	Yes		Yes	Yes	
Maximum instances	MAXINST						Yes			V 9.0.0 Yes
Maximum instances per client	MAXINSTC						Yes			
Maximum message length	MAXMSGL	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	V 9.0.0 Yes
Message channel agent name	MCANAME	Yes	Yes		Yes			Yes	Yes	
Message channel agent type	MCATYPE	Yes	Yes		Yes			Yes	Yes	
Message channel agent user	MCAUSER	Yes	Yes	Yes	Yes		Yes	Yes	Yes	V 9.0.0 Yes
Message exit name	MSGEXIT	Yes	Yes	Yes	Yes			Yes	Yes	
Message exit user data	MSGDATA	Yes	Yes	Yes	Yes			Yes	Yes	
Message-retry exit name	MREXIT			Yes	Yes				Yes	
Message-retry exit user data	MRDATA			Yes	Yes				Yes	
Message retry count	MRRTY			Yes	Yes				Yes	
Message retry interval	MRTMR			Yes	Yes				Yes	
Monitoring	MONCHL	Yes	Yes	Yes	Yes		Yes	Yes	Yes	
Network-connection priority	NETPRTY								Yes	
Nonpersistent message speed	NPMSPEED	Yes	Yes	Yes	Yes			Yes	Yes	
Password	PASSWORD	Yes	Yes		Yes	Yes		Yes		
V 9.0.0 V 9.0.0 Port number	PORT									V 9.0.0 Yes
Property control	PROPCTL	Yes	Yes					Yes	Yes	

Table 28. Channel attributes for the channel types (continued)										
Attribute field	**MQSC command parameter**	**SDR**	**SVR**	**RCVR**	**RQSTR**	**CLNT-CONN**	**SVR-CONN**	**CLUS-SDR**	**CLUS-RCVR**	**V 9.0.0 AMQP**
PUT authority	PUTAUT			Yes	Yes		Yes "2" on page 104		Yes	
Queue manager name	QMNAME					Yes				
Receive exit name	RCVEXIT	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	
Receive exit user data	RCVDATA	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	
Security exit name	SCYEXIT	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	
Security exit user data	SCYDATA	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	
Send exit name	SENDEXIT	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	
Send exit user data	SENDDATA	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	
Sequence number wrap	SEQWRAP	Yes	Yes	Yes	Yes			Yes	Yes	
Shared connections	SHARECNV					Yes	Yes			
Short retry count	SHORTRTY	Yes	Yes					Yes	Yes	
Short retry interval	SHORTTMR	Yes	Yes					Yes	Yes	
SSL Cipher Specification	SSLCIPH	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	V 9.0.0 Yes
SSL Client Authentication	SSLCAUTH		Yes	Yes	Yes		Yes		Yes	V 9.0.0 Yes
SSL Peer	SSLPEER	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	V 9.0.0 Yes
V 9.0.0 V 9.0.0 Topic root	TPROOT									V 9.0.0 Yes
Transmission queue name	XMITQ	Yes	Yes							
Transport type	TRPTYPE	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	
V 9.0.0 V 9.0.0 Use client ID	USECLTID									V 9.0.0 Yes

Table 28. Channel attributes for the channel types (continued)										
Attribute field	**MQSC command parameter**	**SDR**	**SVR**	**RCVR**	**RQSTR**	**CLNT-CONN**	**SVR-CONN**	**CLUS-SDR**	**CLUS-RCVR**	**V 9.0.0 AMQP**
Use Dead-Letter Queue	USEDLQ	Yes	Yes	Yes	Yes			Yes	Yes	
User ID	USERID	Yes	Yes		Yes	Yes		Yes		

Notes:

1. None of the administrative interfaces allow this attribute to be inquired or set for CLUSSDR channels. You will receive an `MQRCCF_WRONG_CHANNEL_TYPE` message. However, the attribute is present in CLUSSDR channel objects (including MQCD structures) and a CHAD exit can set it programmatically if required.
2. z/OS Valid on z/OS only.

Related concepts

"Channel attributes in alphabetical order" on page 104
This section describes each attribute of a channel object, with its valid values and notes on its use where appropriate.

Related reference

MQSC reference

Channel attributes in alphabetical order

This section describes each attribute of a channel object, with its valid values and notes on its use where appropriate.

IBM MQ for some platforms might not implement all the attributes shown in this section. Exceptions and platform differences are mentioned in the individual attribute descriptions, where relevant.

The keyword that you can specify in MQSC is shown in brackets for each attribute.

The attributes are arranged in alphabetical order.

Alter date (ALTDATE)

This attribute is the date on which the definition was last altered, in the form `yyyy-mm-dd`.

This attribute is valid for all channel types.

Alter time (ALTTIME)

This attribute is the time at which the definition was last altered, in the form `hh:mm:ss`.

This attribute is valid for all channel types.

Multi AMQP keep alive (AMQPKA)

Use the **AMQPKA** attribute to specify a keep alive time for the AMQP client connection. If the AMQP client has not sent any frames within the keep alive interval, then the connection is closed.

The **AMQPKA** attribute determines the value of the idle-timeout attribute sent from IBM MQ to an AMQP client. The attribute is a period of time in milliseconds.

If **AMQPKA** is set to a value > 0, then IBM MQ flows half that value as the idle-timeout attribute. For example, a value of 10000 causes the queue manager to send an idle-timeout value of 5000. The client should ensure that data is sent to IBM MQ at least every 10000 milliseconds. If data is not received

by IBM MQ in that time, IBM MQ assumes that the client has lost its connection and forcibly closes the connection with an `amqp:resource-limit-exceeded` error condition.

A value of AUTO or 0 means the IBM MQ does not flow an idle-timeout attribute to the AMQP client.

An AMQP client can still flow an idle-timeout value of its own. If it does, IBM MQ flows data (or an empty AMQP frame) at least that frequently to inform the client that it is available.

Batch Heartbeat Interval (BATCHHB)

This attribute allows a sending channel to verify that the receiving channel is still active just before committing a batch of messages.

The batch heartbeat interval thus allows the batch to be backed out rather than becoming in-doubt if the receiving channel is not active. By backing out the batch, the messages remain available for processing so they could, for example, be redirected to another channel.

If the sending channel has had a communication from the receiving channel within the batch heartbeat interval, the receiving channel is assumed to be still active, otherwise a 'heartbeat' is sent to the receiving channel to check. The sending channel waits for a response from the receiving end of the channel for an interval, based on the number of seconds specified in the channel Heartbeat Interval (HBINT) attribute.

The value is in milliseconds and must be in the range zero through 999999. A value of zero indicates that batch heart beating is not used.

This attribute is valid for channel types of:

- Sender
- Server
- Cluster sender
- Cluster receiver

Batch interval (BATCHINT)

This attribute is a period, in milliseconds, during which the channel keeps a batch open even if there are no messages on the transmission queue.

You can specify any number of milliseconds, from zero through 999 999 999. The default value is zero.

If you do not specify a batch interval, the batch closes when one of the following conditions is met:

- The number of messages specified in BATCHSZ have been sent.
- The number of bytes specified in BATCHLIM have been sent.
- The transmission queue is empty.

On channels with a light load, where the transmission queue frequently becomes empty, the effective batch size might be much smaller than BATCHSZ.

You can use the BATCHINT attribute to make your channels more efficient by reducing the number of short batches. Be aware, however, that you can slow down the response time, because batches last longer and messages remain uncommitted for longer.

If you specify a BATCHINT, batches close only when one of the following conditions is met:

- The number of messages specified in BATCHSZ have been sent.
- The number of bytes specified in BATCHLIM have been sent.
- There are no more messages on the transmission queue and a time interval of BATCHINT has elapsed while waiting for messages (since the first message of the batch was retrieved).

Note: BATCHINT specifies the total amount of time that is spent waiting for messages. It does not include the time spent retrieving messages that are already available on the transmission queue, or the time spent transferring messages.

This attribute is valid for channel types of:

- Sender
- Server
- Cluster sender
- Cluster receiver

Batch limit (BATCHLIM)

This attribute is the limit, in kilobytes, of the amount of data that can be sent through a channel before taking a sync point.

A sync point is taken after the message that caused the limit to be reached has flowed across the channel.

The value must be in the range 0 - 999999. The default value is 5000.

A value of zero in this attribute means that no data limit is applied to batches over this channel.

The batch is terminated when one of the following conditions is met:

- BATCHSZ messages have been sent.
- BATCHLIM bytes have been sent.
- The transmission queue is empty and BATCHINT is exceeded.

This attribute is valid for channel types of:

- Sender
- Server
- Cluster sender
- Cluster receiver

This parameter is supported on all platforms.

Batch size (BATCHSZ)

This attribute is the maximum number of messages to be sent before a sync point is taken.

The batch size does not affect the way the channel transfers messages; messages are always transferred individually, but are committed or backed out as a batch.

To improve performance, you can set a batch size to define the maximum number of messages to be transferred between two *sync points*. The batch size to be used is negotiated when a channel starts, and the lower of the two channel definitions is taken. On some implementations, the batch size is calculated from the lowest of the two channel definitions and the two queue manager MAXUMSGS values. The actual size of a batch can be less; for example, a batch completes when there are no messages left on the transmission queue or the batch interval expires.

A large value for the batch size increases throughput, but recovery times are increased because there are more messages to back out and send again. The default BATCHSZ is 50, and you are advised to try that value first. You might choose a lower value for BATCHSZ if your communications are unreliable, making the need to recover more likely.

Sync point procedure needs a unique logical unit of work identifier to be exchanged across the link every time a sync point is taken, to coordinate batch commit procedures.

If the synchronized batch commit procedure is interrupted, an *in-doubt* situation might arise. In-doubt situations are resolved automatically when a message channel starts. If this resolution is not successful, manual intervention might be necessary, using the RESOLVE command.

Some considerations when choosing the number for batch size:

- If the number is too large, the amount of queue space taken up on both ends of the link becomes excessive. Messages take up queue space when they are not committed, and cannot be removed from queues until they are committed.

- If there is likely to be a steady flow of messages, you can improve the performance of a channel by increasing the batch size because fewer confirm flows are needed to transfer the same quantity of bytes.
- If message flow characteristics indicate that messages arrive intermittently, a batch size of 1 with a relatively large disconnect time interval might provide a better performance.
- The number can be in the range 1 through 9999.
- Even though nonpersistent messages on a fast channel do not wait for a sync point, they do contribute to the batch-size count.

This attribute is valid for channel types of:

- Sender
- Server
- Receiver
- Requester
- Cluster sender
- Cluster receiver

Certificate label (CERTLABL)

This attribute specifies the certificate label of the channel definition.

The label identifies which personal certificate in the key repository is sent to the remote peer. The certificate is defined as described in Digital certificate labels.

Inbound channels (including RCVR, RQSTR, CLUSRCVR, unqualified SERVER, and SVRCONN channels) will only send the configured certificate if the IBM MQ version of the remote peer fully supports certificate label configuration and the channel is using a TLS CipherSpec. If that is not the case, the queue manager **CERTLABL** attribute determines the certificate sent. This restriction is because the certificate label selection mechanism for inbound channels depends upon a TLS protocol extension that is not supported in all cases. In particular, Java clients and JMS clients, do not support the required protocol extension and will only ever receive the certificate configured by the queue manager **CERTLABL** attribute, regardless of the channel-specific label setting.

An unqualified server channel is one that does not have the CONNAME field set.

None of the administrative interfaces allow this attribute to be inquired or set for CLUSSDR channels. You will receive an MQRCCF_WRONG_CHANNEL_TYPE message. However, the attribute is present in CLUSSDR channel objects (including MQCD structures) and a CHAD exit can set it programmatically if required.

For more information about what the certificate label can contain, see Digital certificate labels, understanding the requirements.

This attribute is valid for all channel types.

Note: For SSL/TLS, the CERTLABL must be defined on the QMGR definition. You can, optionally, set a CERTLABL on the CHANNEL definition, however, channels continue to use the queue manager default CERTLABL even if you have defined CERTLABL as a channel attribute.

The queue manager CERTLABL is checked and must be a valid personal certificate, even if you are setting a CERTLABL on the CHANNEL definition.

Channels continue to use the queue manager default CERTLABL, even if you have specified OPMODE in the CSQ6SYSP module.

Channel name (CHANNEL)

This attribute specifies the name of the channel definition.

The name can contain up to 20 characters, although as both ends of a message channel must have the same name, and other implementations might have restrictions on the size, the actual number of characters might have to be smaller.

Where possible, channel names are unique to one channel between any two queue managers in a network of interconnected queue managers.

The name must contain characters from the following list:

Alphabetic	(A-Z, a-z; note that uppercase and lowercase are significant)
Numerics	(0-9)
Period	(.)
Forward slash	(/)
Underscore	(_)
Percentage sign	(%)

Note:

1. Embedded blanks are not allowed, and leading blanks are ignored.
2. On systems using EBCDIC Katakana, you cannot use lowercase characters.

This attribute is valid for all channel types.

Multi Channel statistics (STATCHL) on Multiplatforms

On Multiplatforms, this attribute controls the collection of statistics data for channels.

The possible values are:

QMGR
Statistics data collection for this channel is based upon the setting of the queue manager attribute STATCHL. This value is the default value.

OFF
Statistics data collection for this channel is disabled.

LOW
Statistics data collection for this channel is enabled with a low ratio of data collection.

MEDIUM
Statistics data collection for this channel is enabled with a moderate ratio of data collection.

HIGH
Statistics data collection for this channel is enabled with a high ratio of data collection.

For more information about channel statistics, see Monitoring reference.

z/OS On z/OS systems, enabling this parameter simply turns on statistics data collection, regardless of the value you select. Specifying `LOW`, `MEDIUM`, or `HIGH` makes no difference to your results. This parameter must be enabled in order to collect channel accounting records.

This attribute is valid for channel types of:

- Sender
- Server
- Receiver
- Requester
- Cluster sender

- Cluster receiver

Channel type (CHLTYPE)

This attribute specifies the type of the channel being defined.

The possible channel types are:

Message channel types:

- Sender
- Server
- Receiver
- Requester
- Cluster-sender
- Cluster-receiver

MQI channel types:

- Client-connection (Windows and UNIX only)

  Note: Client-connection channels can also be defined on z/OS for use on other platforms.
- Server-connection
- AMQP

The two ends of a channel must have the same name and have compatible types:

- Sender with receiver
- Requester with server
- Requester with sender (for callback)
- Server with receiver (server is used as a sender)
- Client-connection with server-connection
- Cluster-sender with cluster-receiver
- AMQP with AMQP

Client channel weight (CLNTWGHT)

This attribute specifies a weighting to influence which client-connection channel definition is used.

The client channel weighting attribute is used so that client channel definitions can be selected at random based on their weighting when more than one suitable definition is available.

When a client issues an MQCONN requesting connection to a queue manager group, by specifying a queue manager name starting with an asterisk, which enables client weight balancing across several queue managers, and more than one suitable channel definition is available in the client channel definition table (CCDT), the definition to use is randomly selected based on the weighting, with any applicable CLNTWGHT(0) definitions selected first in alphabetical order.

Specify a value in the range 0 - 99. The default is 0.

A value of 0 indicates that no load balancing is performed and applicable definitions are selected in alphabetical order. To enable load balancing choose a value in the range 1 - 99 where 1 is the lowest weighting and 99 is the highest. The distribution of connections between two or more channels with non-zero weightings is proportional to the ratio of those weightings. For example, three channels with CLNTWGHT values of 2, 4, and 14 are selected approximately 10%, 20%, and 70% of the time. This distribution is not guaranteed. If the AFFINITY attribute of the connection is set to PREFERRED, the first connection chooses a channel definition according to client weightings, and then subsequent connections continue to use the same channel definition.

This attribute is valid for the client-connection channel type only.

Cluster (CLUSTER)

This attribute is the name of the cluster to which the channel belongs.

The maximum length is 48 characters conforming to the rules for naming IBM MQ objects.

Up to one of the resultant values of CLUSTER or CLUSNL can be non-blank. If one of the values is non-blank, the other must be blank.

This attribute is valid for channel types of:

- Cluster sender
- Cluster receiver

Cluster namelist (CLUSNL)

This attribute is the name of the namelist that specifies a list of clusters to which the channel belongs.

Up to one of the resultant values of CLUSTER or CLUSNL can be nonblank. If one of the values is nonblank, the other must be blank.

This attribute is valid for channel types of:

- Cluster sender
- Cluster receiver

CLWLPRTY channel attribute

The CLWLPRTY channel attribute specifies the priority order for channels for cluster workload distribution. The value must be in the range 0-9, where 0 is the lowest priority and 9 is the highest.

Use the CLWLPRTY channel attribute to set a priority order for the available cluster destinations. IBM MQ selects the destinations with the highest priority before selecting destinations with the lowest cluster destination priority. If there are multiple destinations with the same priority, it selects the least recently used destination.

If there are two possible destinations, you can use this attribute to allow failover. Messages go to the queue manager with the highest priority channel. If it becomes unavailable then messages go to the next highest priority queue manager. Lower priority queue managers act as reserves.

IBM MQ checks channel status before prioritizing the channels. Only available queue managers are candidates for selection.

Notes:

- Specify this attribute on the cluster-receiver channel at the target queue manager. Any balancing you specify on the matching cluster-sender channel is likely to be ignored. See Cluster channels.
- 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.

CLWLRANK channel attribute

The **CLWLRANK** channel attribute specifies the rank of channels for cluster workload distribution. The value must be in the range 0-9, where 0 is the lowest rank and 9 is the highest.

Use the **CLWLRANK** channel attribute if you want control over the final destination for messages sent to a queue manager in another cluster. Control the choice of final destination by setting the rank of the channels connecting a queue manager to the gateway queue managers at the intersection of the clusters.

When you set **CLWLRANK**, messages take a specified route through the interconnected clusters towards a higher ranked destination. For example, messages arrive at a gateway queue manager that can send them to either of two queue managers using channels ranked 1 and 2. They are automatically sent to the queue manager connected by a channel with the highest rank, in this case the channel to the queue manager ranked 2.

IBM MQ gets the rank of channels before checking channel status. Getting the rank before checking channel status means that even non-accessible channels are available for selection. It allows messages to be routed through the network even if the final destination is unavailable.

Notes:

- Specify this attribute on the cluster-receiver channel at the target queue manager. Any balancing you specify on the matching cluster-sender channel is likely to be ignored. See Cluster channels.
- If you also used the priority attribute **CLWLPRTY**, IBM MQ selects between available destinations. If a channel is not available to the destination with the highest rank, the message is held on the transmission queue. It is released when the channel becomes available. The message does not get sent to the next available destination in the rank order.

CLWLWGHT channel attribute

The CLWLWGHT channel attribute specifies the weight applied to CLUSSDR and CLUSRCVR channels for cluster workload distribution. The value must be in the range 1-99, where 1 is the lowest weight and 99 is the highest.

Use CLWLWGHT to send servers with more processing power more messages. The higher the channel weight, the more messages are sent over that channel.

Notes:

- Specify this attribute on the cluster-receiver channel at the target queue manager. Any balancing you specify on the matching cluster-sender channel is likely to be ignored. See Cluster channels.
- When CLWLWGHT is modified from the default of 50 on any channel, workload balancing becomes dependent on the total number of times each channel was chosen for a message sent to any clustered queue. For more information, see “The cluster workload management algorithm” on page 153.

Connection affinity (AFFINITY)

This attribute specifies whether client applications that connect multiple times using the same queue manager name, use the same client channel.

Use this attribute when multiple applicable channel definitions are available.

The possible values are:

PREFERRED

The first connection in a process reading a client channel definition table (CCDT) creates a list of applicable definitions based on the client channel weight, with any definitions having a weight of 0 first and in alphabetical order. Each connection in the process attempts to connect using the first definition in the list. If a connection is unsuccessful the next definition is used. Unsuccessful definitions with client channel weight values other than 0 are moved to the end of the list. Definitions with a client channel weight of 0 remain at the start of the list and are selected first for each connection.

Each client process with the same host name always creates the same list.

For client applications written in C, C++, or the .NET programming framework (including fully managed .NET), and for applications that use the IBM MQ classes for Java and IBM MQ classes for JMS, the list is updated if the CCDT has been modified since the list was created.

This value is the default value.

NONE

The first connection in a process reading a CCDT creates a list of applicable definitions. All connections in a process select an applicable definition based on the client channel weight, with any definitions having a weight of 0 selected first in alphabetical order.

For client applications written in C, C++, or the .NET programming framework (including fully managed .NET), and for applications that use the IBM MQ classes for Java and IBM MQ classes for JMS, the list is updated if the CCDT has been modified since the list was created.

This attribute is valid for the client-connection channel type only.

Connection name (CONNAME)

This attribute is the communications connection identifier. It specifies the particular communications links to be used by this channel.

It is optional for server channels, unless the server channel is triggered, in which case it must specify a connection name.

Specify **CONNAME** as a comma-separated list of names of machines for the stated **TRPTYPE**. Typically only one machine name is required. You can provide multiple machine names to configure multiple connections with the same properties. The connections are usually tried in the order they are specified in the connection list until a connection is successfully established. The order is modified for clients if the **CLNTWGHT** attribute is provided. If no connection is successful, the channel attempts the connection again, as determined by the attributes of the channel. With client channels, a connection-list provides an alternative to using queue manager groups to configure multiple connections. With message channels, a connection list is used to configure connections to the alternative addresses of a multi-instance queue manager.

Providing multiple connection names in a list was first supported in IBM WebSphere® MQ 7.0.1. It changes the syntax of the **CONNAME** parameter. Earlier clients and queue managers connect using the first connection name in the list, and do not read the rest of the connection names in the list. In order for the earlier clients and queue managers to parse the new syntax, you must specify a port number on the first connection name in the list. Specifying a port number avoids problems when connecting to the channel from a client or queue manager that is running at a level earlier than IBM WebSphere MQ 7.0.1.

Multi On Multiplatforms, the TCP/IP connection name parameter of a cluster-receiver channel is optional. If you leave the connection name blank, IBM MQ generates a connection name for you, assuming the default port and using the current IP address of the system. You can override the default port number, but still use the current IP address of the system. For each connection name leave the IP name blank, and provide the port number in parentheses; for example:

```
(1415)
```

The generated **CONNAME** is always in the dotted decimal (IPv4) or hexadecimal (IPv6) form, rather than in the form of an alphanumeric DNS host name.

The maximum name length depends on the platform:

- Multi 264 characters.
- z/OS 48 characters (see note 1).

If the transport type is TCP

CONNAME is either the host name or the network address of the remote machine (or the local machine for cluster-receiver channels). For example, `(ABC.EXAMPLE.COM)`, `(2001:DB8:0:0:0:0:0:0)` or `(127.0.0.1)`. It can include the port number, for example `(MACHINE(123))`.

z/OS It can include the IP_name of a dynamic DNS group or a Network Dispatcher input port.

If you use an IPv6 address in a network that only supports IPv4, the connection name is not resolved. In a network which uses both IPv4 and IPv6, the connection name interacts with the local address to determine which IP stack is used. See “Local Address (LOCLADDR)” on page 118 for further information.

If the transport type is LU 6.2

Windows IBM i UNIX If the TPNAME and MODENAME are specified, give the fully-qualified name of the partner LU.

Multi If the TPNAME and MODENAME are blank, give the CPI-C side information object name for your specific platform.

z/OS There are two forms in which to specify the value:

- Logical unit name

  The logical unit information for the queue manager, comprising the logical unit name, TP name, and optional mode name. This name can be specified in one of three forms:

Form	Example
luname	`IGY12355`
luname/TPname	`IGY12345/APING`
luname/TPname/modename	`IGY12345/APINGD/#INTER`

  For the first form, the TP name and mode name must be specified for the TPNAME and MODENAME attributes; otherwise these attributes must be blank. For client-connection channels, only the first form is allowed.

- Symbolic name

  The symbolic destination name for the logical unit information for the queue manager, as defined in the side information data set. The TPNAME and MODENAME attributes must be blank. Note that, for cluster-receiver channels, the side information is on the other queue managers in the cluster. In this case it can be a name that a channel auto-definition exit can resolve into the appropriate logical unit information for the local queue manager.

The specified or implied LU name can be that of a VTAM generic resources group.

If the transmission protocol is NetBIOS

CONNAME is the NetBIOS name defined on the remote machine.

If the transmission protocol is SPX

CONNAME is an SPX-style address consisting of a 4 byte network address, a 6 byte node address and a 2 byte socket number. Enter these values in hexadecimal, with the network and node addresses separated by a period and the socket number in brackets. For example:

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

If the socket number is omitted, the default IBM MQ SPX socket number is used. The default is X'5E86'.

This attribute is valid for channel types of:

- Sender
- Server
- Requester
- Client connection
- Cluster sender
- Cluster receiver

It is optional for server channels, unless the server channel is triggered, in which case it must specify a connection name.

Note:

1. For name lengths, you can work around the 48 character limit in either of the following ways:
   - Set up your DNS servers so that you use, for example, host name of "myserver" instead of "myserver.location.company.com", ensuring you can use the short host name.
   - Use IP addresses.
2. The definition of transmission protocol is contained in “Transport type (TRPTYPE)” on page 135.

Convert message (CONVERT)

This attribute specifies that the message must be converted into the format required by the receiving system before transmission.

Application message data is typically converted by the receiving application. However, if the remote queue manager is on a platform that does not support data conversion, use this channel attribute to specify that the message must be converted into the format required by the receiving system ***before*** transmission.

The possible values are yes and no. If you specify yes, the application data in the message is converted before sending if you have specified one of the built-in format names, or a data conversion exit is available for a user-defined format (See Writing data-conversion exits). If you specify no, the application data in the message is not converted before sending.

This attribute is valid for channel types of:

- Sender
- Server
- Cluster sender
- Cluster receiver

Data compression (COMPMSG)

This attribute is a list of message data compression techniques supported by the channel.

For sender, server, cluster-sender, cluster-receiver, and client-connection channels the values specified are in order of preference. The first compression technique supported by the remote end of the channel is used. The channels' mutually supported compression techniques are passed to the sending channel's message exit where the compression technique used can be altered on a per message basis. Compression alters the data passed to send and receive exits. See “Header compression (COMPHDR)” on page 116 for compression of the message header.

The possible values are:

NONE
No message data compression is performed. This value is the default value.

RLE
Message data compression is performed using run-length encoding.

ZLIBFAST
Message data compression is performed using the zlib compression technique. A fast compression time is preferred.

ZLIBFAST can optionally be offloaded to the zEnterprise® Data Compression facility. See zEDC Express facility for further information.

ZLIBHIGH
Message data compression is performed using the zlib compression technique. A high level of compression is preferred.

ANY
Allows the channel to support any compression technique that the queue manager supports. Only supported for Receiver, Requester and Server-Connection channels.

This attribute is valid for all channel types.

Default reconnection (DEFRECON)

Specifies whether a client connection automatically reconnects a client application if its connection breaks.

The possible values are:

NO (default)
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.

This attribute is valid only for client connection channels.

Description (DESCR)

This attribute describes the channel definition and contains up to 64 bytes of text.

Note: The maximum number of characters is reduced if the system is using a double byte character set (DBCS).

Use characters from the character set identified by the coded character set identifier (CCSID) for the queue manager to ensure that the text is translated correctly if it is sent to another queue manager.

This attribute is valid for all channel types.

Disconnect interval (DISCINT)

This attribute is the length of time after which a channel closes down, if no message arrives during that period.

This attribute is a time-out attribute, specified in seconds, for the server, cluster-sender, sender, and cluster-receiver channels. The interval is measured from the point at which a batch ends, that is when the batch size is reached or when the batch interval expires and the transmission queue becomes empty. If no messages arrive on the transmission queue during the specified time interval, the channel closes down. (The time is approximate.)

The close-down exchange of control data between the two ends of the channel includes an indication of the reason for closing. This ensures that the corresponding end of the channel remains available to start again.

You can specify any number of seconds from zero through 999 999 where a value of zero means no disconnect; wait indefinitely.

For server-connection channels using the TCP protocol, the interval represents the client inactivity disconnect value, specified in seconds. If a server-connection has received no communication from its partner client for this duration, it terminates the connection.

The server-connection inactivity interval applies between IBM MQ API calls from a client.

Note: A potentially long-running MQGET with wait call is not classified as inactivity and, therefore, never times out as a result of DISCINT expiring.

This attribute is valid for channel types of:

- Sender
- Server
- Server connection
- Cluster sender
- Cluster receiver

This attribute is not applicable for server-connection channels using protocols other than TCP.

Note: Performance is affected by the value specified for the disconnect interval.

A low value (for example a few seconds) can be detrimental to system performance by constantly starting the channel. A large value (more than an hour) might mean that system resources are needlessly held up. You can also specify a heartbeat interval, so that when there are no messages on the transmission queue, the sending MCA sends a heartbeat flow to the receiving MCA, thus giving the receiving MCA an opportunity to quiesce the channel without waiting for the disconnect interval to expire. For these two values to work together effectively, the heartbeat interval value must be significantly lower than the disconnect interval value.

The default DISCINT value is set to 100 minutes. However, a value of a few minutes is often a reasonable value to use without impacting performance or keeping channels running for unnecessarily long periods of time. If it is appropriate for your environment you can change this value, either on each individual channel or through changing the value in the default channel definitions, for example SYSTEM.DEF.SENDER.

For more information, see Stopping and quiescing channels.

Disposition (QSGDISP)

This attribute specifies the disposition of the channel in a queue sharing group. It is valid on z/OS only.

Values are:

QMGR
The channel is defined on the page set of the queue manager that executes the command. This value is the default.

GROUP
The channel is defined in the shared repository. This value is allowed only if there is a shared queue manager environment. When a channel is defined with QSGDISP(GROUP), the command DEFINE CHANNEL(name) NOREPLACE QSGDISP(COPY) is generated automatically and sent to all active queue managers to cause them to make local copies on page set 0. For queue managers which are not active, or which join the queue sharing group at a later date, the command is generated when the queue manager starts.

COPY
The channel is defined on the page set of the queue manager that executes the command, copying its definition from the QSGDISP(GROUP) channel of the same name. This value is allowed only if there is a shared queue manager environment.

This attribute is valid for all channel types.

Header compression (COMPHDR)

This attribute is a list of header data compression techniques supported by the channel.

For sender, server, cluster-sender, cluster-receiver, and client-connection channels the values specified are in order of preference with the first compression technique supported by the remote end of the channel being used. The channels' mutually supported compression techniques are passed to the sending channel's message exit where the compression technique used can be altered on a per message basis. Compression alters the data passed to send and receive exits.

Possible values are:

NONE
No header data compression is performed. This value is the default value.

SYSTEM
Header data compression is performed.

This attribute is valid for all channel types.

Heartbeat interval (HBINT)

This attribute specifies the approximate time between heartbeat flows that are to be passed from a sending message channel agent (MCA) when there are no messages on the transmission queue.

Heartbeat flows unblock the receiving MCA, which is waiting for messages to arrive or for the disconnect interval to expire. When the receiving MCA is unblocked, it can disconnect the channel without waiting for the disconnect interval to expire. Heartbeat flows also free any storage buffers that have been allocated for large messages and close any queues that have been left open at the receiving end of the channel.

The value is in seconds and must be in the range 0 - 999 999. A value of zero means that no heartbeat flows are to be sent. The default value is 300. To be most useful, the value must be significantly less than the disconnect interval value.

With applications that use IBM MQ classes for Java, JMS or .NET APIs, the HBINT value is determined in one of the following ways:

- Either by the value on the SVRCONN channel that is used by the application.
- Or by the value on the CLNTCONN channel, if the application has been configured to use a CCDT.

For server-connection and client-connection channels, heartbeats can flow from both the server side as well as the client side independently. If no data has been transferred across the channel for the heartbeat interval, the client-connection MQI agent sends a heartbeat flow and the server-connection MQI agent responds to it with another heartbeat flow. This happens irrespective of the state of the channel, for example, irrespective of whether it is inactive while making an API call, or is inactive waiting for client user input. The server-connection MQI agent is also capable of initiating a heartbeat to the client, again irrespective of the state of the channel. To prevent both server-connection and client-connection MQI agents heart beating to each other at the same time, the server heartbeat is flowed after no data has been transferred across the channel for the heartbeat interval plus 5 seconds.

For server-connection and client-connection channels working in the channel mode before IBM WebSphere MQ 7.0, heartbeats flow only when a server MCA is waiting for an MQGET command with the WAIT option specified, which it has issued on behalf of a client application.

For more information about making MQI channels work in the two modes, see SharingConversations (MQLONG).

Related reference
DEFINE CHANNEL
ALTER CHANNEL

Keepalive Interval (KAINT)

This attribute is used to specify a timeout value for a channel.

The Keepalive Interval attribute is a value passed to the communications stack specifying the Keepalive timing for the channel. It allows you to specify a different keepalive value for each channel.

You can set the Keepalive Interval (KAINT) attribute for channels on a per-channel basis.

Multi On Multiplatforms, you can access and modify the parameter, but it is only stored and forwarded; there is no functional implementation of the parameter. If you need the functionality provided by the KAINT parameter, use the Heartbeat Interval (HBINT) parameter, as described in “Heartbeat interval (HBINT)” on page 117.

For this attribute to have any effect, TCP/IP keepalive must be enabled. On z/OS, you do enable keepalive by issuing the ALTER QMGR TCPKEEP(YES) MQSC command. On Multiplatforms, it occurs when the KEEPALIVE=YES parameter is specified in the TCP stanza in the distributed queuing configuration file, qm.ini, or through the IBM MQ Explorer. Keepalive must also be enabled within TCP/IP itself, using the TCP profile configuration data set.

The value indicates a time, in seconds, and must be in the range 0 - 99999. A Keepalive Interval value of 0 indicates that channel-specific Keepalive is not enabled for the channel and only the system-wide Keepalive value set in TCP/IP is used. You can also set KAINT to a value of AUTO (this value is the default). If KAINT is set to AUTO, the Keepalive value is based on the value of the negotiated heartbeat interval (HBINT) as follows:

Table 29. Negotiated HBINT value and the corresponding KAINT value.

The table has two columns. The first column lists the negotiated HBINT values and the second column lists the corresponding KAINT value for each negotiated HBINT.

Negotiated HBINT	KAINT
>0	Negotiated HBINT + 60 seconds
0	0

This attribute is valid for all channel types.

The value is ignored for all channels that have a TransportType (TRPTYPE) other than TCP or SPX

Local Address (LOCLADDR)

This attribute specifies the local communications address for the channel.

Note: AMQP channels do not support the same format of LOCLADDR as other IBM MQ channels. For more information, see "LOCLADDR for AMQP channels" on page 120.

LOCLADDR for all channels except AMQP channels

This attribute only applies if the transport type (TRPTYPE) is TCP/IP. For all other transport types, it is ignored.

When a LOCLADDR value is specified, a channel that is stopped and then restarted continues to use the TCP/IP address specified in LOCLADDR. In recovery scenarios, this attribute might be useful when the channel is communicating through a firewall. It is useful because it removes problems caused by the channel restarting with the IP address of the TCP/IP stack to which it is connected. LOCLADDR can also force a channel to use an IPv4 or IPv6 stack on a dual stack system, or a dual-mode stack on a single stack system.

This attribute is valid for channel types of:

- Sender
- Server
- Requester
- Client connection
- Cluster sender
- Cluster receiver

When LOCLADDR includes a network address, the address must be a network addresses belonging to a network interface on the system where the channel is run. For example, when defining a sender channel on queue manager ALPHA to queue manager BETA with the following MSQC command:

```
DEFINE CHANNEL(TO.BETA) CHLTYPE(SDR) CONNAME(192.0.2.0) XMITQ(BETA) LOCLADDR(192.0.2.1)
```

The LOCLADDR address is the IPv4 address 192.0.2.1. This sender channel runs on the system of queue manager ALPHA, so the IPv4 address must belong to one of the network interfaces its system.

The value is the optional IP address, and optional port or port range used for outbound TCP/IP communications. The format for this information is as follows:

```
LOCLADDR([ip-addr][(low-port[,high-port])][,[ip-addr][(low-port[,high-port])]])
```

The maximum length of **LOCLADDR**, including multiple addresses, is MQ_LOCAL_ADDRESS_LENGTH.

If you omit **LOCLADDR**, a local address is automatically allocated.

Note, that you can set **LOCLADDR** for a C client using the Client Channel Definition Table (CCDT).

All the parameters are optional. Omitting the ip-addr part of the address is useful to enable the configuration of a fixed port number for an IP firewall. Omitting the port number is useful to select a particular network adapter without having the identify a unique local port number. The TCP/IP stack generates a unique port number.

Specify *[,[ip-addr][(low-port[,high-port])]]* multiple times for each additional local address. Use multiple local addresses if you want to specify a specific subset of local network adapters. You can also use *[,[ip-addr][(low-port[,high-port])]]* to represent a particular local network address on different servers that are part of a multi-instance queue manager configuration.

ip-addr
: ip-addr is specified in one of three forms:

  IPv4 dotted decimal
  : For example, 192.0.2.1

  IPv6 hexadecimal notation
  : For example, 2001:DB8:0:0:0:0:0:0

  Alphanumeric host name form
  : For example WWW.EXAMPLE.COM

low-port and high-port
: low-port and high-port are port numbers enclosed in parentheses.

The following table shows how the **LOCLADDR** parameter can be used:

*Table 30. Examples of how the **LOCLADDR** parameter can be used*

LOCLADDR	Meaning
9.20.4.98	Channel binds to this address locally
9.20.4.98, 9.20.4.99	Channel binds to either IP address. The address might be two network adapters on one server, or a different network adapter on two different servers in a multi-instance configuration.
9.20.4.98(1000)	Channel binds to this address and port 1000 locally
9.20.4.98(1000,2000)	Channel binds to this address and uses a port in the range 1000 - 2000 locally
(1000)	Channel binds to port 1000 locally
(1000,2000)	Channel binds to port in range 1000 - 2000 locally

When a channel is started the values specified for connection name (CONNAME) and local address (LOCLADDR) determine which IP stack is used for communication. The IP stack used is determined as follows:

- If the system has only an IPv4 stack configured, the IPv4 stack is always used. If a local address (LOCLADDR) or connection name (CONNAME) is specified as an IPv6 network address, an error is generated and the channel fails to start.

- If the system has only an IPv6 stack configured, the IPv6 stack is always used. If a local address (LOCLADDR) is specified as an IPv4 network address, an error is generated and the channel fails to start. On platforms supporting IPv6 mapped addressing, if a connection name (CONNAME) is specified as an IPv4 network address, the address is mapped to an IPv6 address. For example, `xxx.xxx.xxx.xxx` is mapped to `::ffff:xxx.xxx.xxx.xxx`. The use of mapped addresses might require protocol translators. Avoid the use of mapped addresses where possible.
- If a local address (LOCLADDR) is specified as an IP address for a channel, the stack for that IP address is used. If the local address (LOCLADDR) is specified as a host name resolving to both IPv4 and IPv6 addresses, the connection name (CONNAME) determines which of the stacks is used. If both the local address (LOCLADDR) and connection name (CONNAME) are specified as host names resolving to both IPv4 and IPv6 addresses, the stack used is determined by the queue manager attribute IPADDRV.
- If the system has dual IPv4 and IPv6 stacks configured and a local address (LOCLADDR) is not specified for a channel, the connection name (CONNAME) specified for the channel determines which IP stack to use. If the connection name (CONNAME) is specified as a host name resolving to both IPv4 and IPv6 addresses, the stack used is determined by the queue manager attribute IPADDRV.

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.

Local addresses with cluster sender channels

Cluster sender channels always inherit the configuration of the corresponding cluster receiver channel as defined on the target queue manager. This is true even if there is a locally defined cluster sender channel of the same name, in which case the manual definition is only used for initial communication.

For this reason, it is not possible to depend on the LOCLADDR defined in the cluster receiver channel as it is likely that the IP address is not owned by the system where the cluster senders are created. For this reason, the LOCLADDR on the cluster receiver should not be used unless there is a reason to restrict only the ports but not the IP address for all potential cluster senders, and it is known that those ports are available on all systems where a cluster sender channel may be created.

If a cluster must use LOCLADDR to get the outbound communication channels to bind to a specific IP address, either use a Channel Auto-Definition Exit, or use the default LOCLADDR for the queue manager when possible. When using a channel exit, it forces the LOCLADDR value from the exit into any of the automatically defined CLUSSDR channels.

If using a non-default LOCLADDR for cluster sender channels through the use of an exit or a default value, any matching manually defined cluster sender channel, for example to a full repository queue manager, must also have the LOCLADDR value set to enable initial communication over the channel.

Note: If the operating system returns a bind error for the port supplied in LOCLADDR (or all ports, if a port range is supplied), the channel does not start; the system issues an error message.

LOCLADDR for AMQP channels

AMQP channels support a different format of LOCLADDR than other IBM MQ channels:

LOCLADDR (*ip-addr*)

LOCLADDR is the local communications address for the channel. Use this parameter if you want to force the client to use a particular IP address. LOCLADDR is also useful to force a channel to use an IPv4 or IPv6 address if a choice is available, or to use a particular network adapter on a system with multiple network adapters.

The maximum length of LOCLADDR is MQ_LOCAL_ADDRESS_LENGTH.

If you omit LOCLADDR, a local address is automatically allocated.

ip-addr
ip-addr is a single network address, specified in one of three forms:

IPv4 dotted decimal
For example `192.0.2.1`

IPv6 hexadecimal notation
For example `2001:DB8:0:0:0:0:0:0`

Alphanumeric host name form
For example `WWW.EXAMPLE.COM`

If an IP address is entered, only the address format is validated. The IP address itself is not validated.

Related concepts
Working with auto-defined cluster-sender channels

Long retry count (LONGRTY)

This attribute specifies the maximum number of times that the channel is to try allocating a session to its partner.

The **`long retry count`** attribute can be set from zero through 999 999 999.

This attribute is valid for the following channel types:

- Sender
- Server
- Cluster sender
- Cluster receiver

If the initial allocation attempt fails, the *`short retry count`* number is decremented and the channel retries the remaining number of times. If it still fails, it retries a *`long retry count`* number of times with an interval of *`long retry interval`* between each try. If it is still unsuccessful, the channel closes down. The channel must then be restarted with a command; it is not started automatically by the channel initiator.

On z/OS, a channel cannot enter retry if the maximum number of channels (**MAXCHL**) has been exceeded.

On IBM i, UNIX, and Windows systems, in order for retry to be attempted a channel initiator must be running. The channel initiator must be monitoring the initiation queue specified in the definition of the transmission queue that the channel is using.

If the channel initiator (on z/OS) or the channel (on Multiplatforms) is stopped while the channel is retrying, the *short retry count* and *long retry count* are reset when the channel initiator or the channel is restarted, or when a message is successfully put at the sender channel. However, if the channel initiator (on z/OS) or queue manager (on Multiplatforms) is shut down and restarted, the *short retry count* and *long retry count* are not reset. The channel retains the retry count values it had before the queue manager restart or the message being put.

For IBM i, UNIX, and Windows systems:

1. When a channel goes from RETRYING state to RUNNING state, the *short retry count* and *long retry count* are not reset immediately. They are reset only when the first message flows across the channel successfully after the channel went into RUNNING state, that is; when the local channel confirms the number of messages sent to the other end.
2. The *short retry count* and *long retry count* are reset when the channel is restarted.

Long retry interval (LONGTMR)

This attribute is the approximate interval in seconds that the channel is to wait before retrying to establish connection, during the long retry mode.

The interval between retries can be extended if the channel has to wait to become active.

The channel tries to connect *`long retry count`* number of times at this long interval, after trying the *`short retry count`* number of times at the short retry interval.

This attribute can be set from zero through 999 999.

This attribute is valid for channel types of:

- Sender
- Server
- Cluster sender
- Cluster receiver

LU 6.2 mode name (MODENAME)

This attribute is for use with LU 6.2 connections. It gives extra definition for the session characteristics of the connection when a communication session allocation is performed.

When using side information for SNA communications, the mode name is defined in the CPI-C Communications Side Object or APPC side information, and this attribute must be left blank; otherwise, it must be set to the SNA mode name.

The name must be one to eight alphanumeric characters long.

This attribute is valid for channel types of:

- Sender
- Server
- Requester
- Client connection
- Cluster sender
- Cluster receiver

It is not valid for receiver or server-connection channels.

LU 6.2 transaction program name (TPNAME)

This attribute is for use with LU 6.2 connections. It is the name, or generic name, of the transaction program (MCA) to be run at the far end of the link.

When using side information for SNA communications, the transaction program name is defined in the CPI-C Communications Side Object or APPC side information and this attribute must be left blank. Otherwise, this name is required by sender channels and requester channels.

The name can be up to 64 characters long.

The name must be set to the SNA transaction program name, unless the CONNAME contains a side-object name in which case it must be set to blanks. The actual name is taken instead from the CPI-C Communications Side Object, or the APPC side information data set.

This information is set in different ways on different platforms; see Configuring distributed queuing for more information about setting up communication for your platform.

This attribute is valid for channel types of:

- Sender
- Server
- Requester
- Client connection
- Cluster sender
- Cluster receiver

Maximum instances (MAXINST)

This attribute specifies the maximum number of simultaneous instances of a server-connection channel or AMQP channel that can be started.

See the child topics for information on how the attribute is used for each channel type.

Related concepts

Server-connection channel limits

Related reference

DEFINE CHANNEL

Maximum instances of server-connection channel connections

This attribute specifies the maximum number of simultaneous instances of a sever-connection channel that can be started.

This attribute can be set from zero through 999 999 999. A value of zero indicates that no client connections are allowed on this channel. The default value is 999 999 999.

If the value is reduced so that it is less than the number of instances of the server-connection channel that are currently running, then the running channels are not affected. However, new instances are not able to start until sufficient existing ones have ceased to run.

V 9.0.0 *Maximum instances of AMQP channel connections*

This attribute specifies the maximum number of simultaneous instances of an AMQP channel that can be started.

This attribute can be set from zero through 999 999 999. A value of zero indicates that no client connections are allowed on this channel. The default value is 999 999 999.

If a client attempts to connect, and the number of connected clients has reached MAXINST, the channel closes the connection with a close frame. The close frame contains the following message:

```
amqp:resource-limit-exceeded
```

If a client connects with an ID that is already connected (that is, it performs a client-takeover) the takeover will succeed regardless of whether the number of connected clients has reached MAXINST.

Maximum instances per client (MAXINSTC)

This attribute specifies the maximum number of simultaneous instances of a server-connection channel that can be started from a single client.

This attribute can be set from zero through 999 999 999. A value of zero indicates that no client connections are allowed on this channel. The default value is 999 999 999.

If the value is reduced so that it is less than the number of instances of the server-connection channel that are currently running from individual clients, then the running channels are not affected. However, new instances from those clients are not able to start until sufficient existing ones have ceased to run.

This attribute is valid for server-connection channels only.

Related concepts

Server-connection channel limits

Related reference

DEFINE CHANNEL

Maximum message length (MAXMSGL)

This attribute specifies the maximum length of a message that can be transmitted on the channel.

Multi On IBM MQ for IBM i, UNIX, and Windows systems, specify a value greater than or equal to zero, and less than or equal to the maximum message length for the queue manager. See the MAXMSGL parameter of the ALTER QMGR command in ALTER QMGR for more information.

z/OS On IBM MQ for z/OS, specify a value greater than or equal to zero, and less than or equal to 104 857 600 bytes (that is, 100 MB).

Because various implementations of IBM MQ systems exist on different platforms, the size available for message processing might be limited in some applications. This number must reflect a size that your system can handle without stress. When a channel starts, the lower of the two numbers at each end of the channel is taken.

By adding the digital signature and key to the message, Advanced Message Security increases the length of the message.

Note: You can use a maximum message size of 0 for the channel, which is taken to mean that the size is to be set to the local queue manager maximum value.

This attribute is valid for all channel types.

Message channel agent name (MCANAME)

This attribute is reserved and if specified must only be set to blanks.

Its maximum length is 20 characters.

Message channel agent type (MCATYPE)

This attribute can specify the message channel agent as a *process* or a *thread*.

On IBM MQ for z/OS, it is supported only for channels with a channel type of cluster-receiver.

Advantages of running as a process include:

- Isolation for each channel providing greater integrity
- Job authority specific for each channel
- Control over job scheduling

Advantages of threads include:

- Much reduced use of storage
- Easier configuration by typing on the command line
- Faster execution - it is quicker to start a thread than to instruct the operating system to start a process

For channel types of sender, server, and requester, the default is `process`. For channel types of cluster-sender and cluster-receiver, the default is `thread`. These defaults can change during your installation.

If you specify `process` on the channel definition, a RUNMQCHL process is started. If you specify `thread`, the MCA runs on a thread of the AMQRMPPA process, or of the RUNMQCHI process if MQNOREMPOOL is specified. On the machine that receives the inbound allocates, the MCA runs as a thread if you use RUNMQLSR. It runs as a process if you use **`inetd`**.

On IBM MQ for z/OS, this attribute is supported only for channels with a channel type of cluster-receiver. On other platforms, it is valid for channel types of:

- Sender
- Server
- Requester
- Cluster sender

- Cluster receiver

Message channel agent user identifier (MCAUSER)

This attribute is the user identifier (a string) to be used by the MCA for authorization to access IBM MQ resources.

Note: An alternative way of providing a user ID for a channel to run under is to use channel authentication records. With channel authentication records, different connections can use the same channel while using different credentials. If both MCAUSER on the channel is set and channel authentication records are used to apply to the same channel, the channel authentication records take precedence. The MCAUSER on the channel definition is only used if the channel authentication record uses USERSRC(CHANNEL).

This authorization includes (if PUT authority is DEF) putting the message to the destination queue for receiver or requester channels.

On IBM MQ for Windows, the user identifier can be domain-qualified by using the format, `user@domain`, where the `domain` must be either the Windows systems domain of the local system, or a trusted domain.

If this attribute is blank, the MCA uses its default user identifier. For more information, see DEFINE CHANNEL.

This attribute is valid for channel types of:

- Receiver
- Requester
- Server connection
- Cluster receiver

Related concepts

Channel authentication records

Message exit name (MSGEXIT)

This attribute specifies the name of the user exit program to be run by the channel message exit.

This attribute can be a list of names of programs that are to be run in succession. Leave blank, if no channel message exit is in effect.

The format and maximum length of this attribute depend on the platform, as for "Receive exit name (RCVEXIT)" on page 130.

This attribute is valid for channel types of:

- Sender
- Server
- Receiver
- Requester
- Cluster sender
- Cluster receiver

Message exit user data (MSGDATA)

This attribute specifies user data that is passed to the channel message exits.

You can run a sequence of message exits. The limitations on the user data length and an example of how to specify MSGDATA for more than one exit are as shown for RCVDATA. See "Receive exit user data (RCVDATA)" on page 131.

This attribute is valid for channel types of:

- Sender

- Server
- Receiver
- Requester
- Cluster sender
- Cluster receiver

Message-retry exit name (MREXIT)

This attribute specifies the name of the user exit program to be run by the message-retry user exit.

Leave blank if no message-retry exit program is in effect.

The format and maximum length of the name depend on the platform, as for "Receive exit name (RCVEXIT)" on page 130. However, there can only be one message-retry exit specified

This attribute is valid for channel types of:

- Receiver
- Requester
- Cluster receiver

Message-retry exit user data (MRDATA)

This attribute specifies data passed to the channel message-retry exit when it is called.

This attribute is valid for channel types of:

- Receiver
- Requester
- Cluster receiver

Message retry count (MRRTY)

This attribute specifies the number of times the channel tries to redeliver the message.

This attribute controls the action of the MCA only if the message-retry exit name is blank. If the exit name is not blank, the value of MRRTY is passed to the exit, but the number of attempts made (if any) is controlled by the exit, and not by this attribute.

The value must be in the range 0 - 999 999 999. A value of zero means that no additional attempts are made. The default is 10.

This attribute is valid for channel types of:

- Receiver
- Requester
- Cluster receiver

Message retry interval (MRTMR)

This attribute specifies the minimum interval of time that must pass before the channel can retry the MQPUT operation.

This time interval is in milliseconds.

This attribute controls the action of the MCA only if the message-retry exit name is blank. If the exit name is not blank, the value of MRTMR is passed to the exit for use by the exit, but the retry interval is controlled by the exit, and not by this attribute.

The value must be in the range 0 - 999 999 999. A value of zero means that the retry is performed as soon as possible (if the value of MRRTY is greater than zero). The default is 1000.

This attribute is valid for the following channel types:

- Receiver
- Requester
- Cluster receiver

Monitoring (MONCHL)

This attribute controls the collection of online Monitoring data.

Possible values are:

QMGR
The collection of Online Monitoring Data is inherited from the setting of the MONCHL attribute in the queue manager object. This value is the default value.

OFF
Online Monitoring Data collection for this channel is disabled.

LOW
A low ratio of data collection with a minimal effect on performance. However, the monitoring results shown might not be up to date.

MEDIUM
A moderate ratio of data collection with limited effect on the performance of the system.

HIGH
A high ratio of data collection with the possibility of an effect on performance. However, the monitoring results shown are the most current.

This attribute is valid for channel types of:

- Sender
- Server
- Receiver
- Requester
- Server connection
- Cluster sender
- Cluster receiver

For more information about monitoring data, see Displaying queue and channel monitoring data.

NETPRTY channel attribute

The NETPRTY channel attribute specifies the priority for a CLUSRCVR channel. The value must be in the range 0-9, where 0 is the lowest priority and 9 is the highest.

Use the NETPRTY attribute to make one network the primary network, and another network the backup network. Given a set of equally ranked channels, clustering chooses the path with the highest priority when multiple paths are available.

A typical example of using the NETPRTY channel attribute is to differentiate between networks that have different costs or speeds and connect the same destinations.

Note: Specify this attribute on the cluster-receiver channel at the target queue manager. Any balancing you specify on the matching cluster-sender channel is likely to be ignored. See Cluster channels.

Nonpersistent message speed (NPMSPEED)

This attribute specifies the speed at which nonpersistent messages are to be sent.

Possible values are:

NORMAL
Nonpersistent messages on a channel are transferred within transactions.

FAST
Nonpersistent messages on a channel are not transferred within transactions.

The default is FAST. The advantage of this is that nonpersistent messages become available for retrieval far more quickly. The disadvantage is that because they are not part of a transaction, messages might be lost if there is a transmission failure or if the channel stops when the messages are in transit. See Safety of messages.

Notes:

1. If the active recovery logs for IBM MQ for z/OS are switching and archiving more frequently than expected, given that the messages being sent across a channel are non-persistent, setting NPMSPEED(FAST) on both the sending and receiving ends of the channel can minimize the SYSTEM.CHANNEL.SYNCQ updates.
2. If you are seeing high CPU usage relating to updates to the SYSTEM.CHANNEL.SYNCQ, setting NPMSPEED(FAST) can significantly reduce the CPU usage.

This attribute is valid for channel types of:

- Sender
- Server
- Receiver
- Requester
- Cluster sender
- Cluster receiver

Password (PASSWORD)

This attribute specifies a password that can be used by the MCA when attempting to initiate a secure LU 6.2 session with a remote MCA.

You can specify a password of maximum length 12 characters, although only the first 10 characters are used.

It is valid for channel types of sender, server, requester, or client-connection.

On IBM MQ for z/OS, this attribute is valid only for client connection channels. On other platforms, it is valid for channel types of:

- Sender
- Server
- Requester
- Client connection
- Cluster sender

Port number (PORT)

Specify the port number that is used to connect the AMQP client.

The default port for AMQP 1.0 connections is 5672. If you are already using port 5672, you can specify a different port.

PUT authority (PUTAUT)

This attribute specifies the type of security processing to be carried out by the MCA.

This attribute is valid for channel types of:

- Receiver

- Requester
- Server connection (z/OS only)
- Cluster receiver

Use this attribute to choose the type of security processing to be carried out by the MCA when executing:

- An MQPUT command to the destination queue (for message channels), or
- An MQI call (for MQI channels).

z/OS On z/OS, the user IDs that are checked, and how many user IDs are checked, depends on the setting of the MQADMIN RACF® class hlq.RESLEVEL profile. Depending on the level of access the user ID of the channel initiator has to hlq.RESLEVEL, zero, one or two user IDs are checked. To see how many user IDs are checked, see RESLEVEL and channel initiator connections. For more information about which user IDs are checked, see User IDs used by the channel initiator.

You can choose one of the following:

Process security, also called default authority (DEF)
The default user ID is used.

On platforms other than z/OS, the user ID used to check open authority on the queue is that of the process or user running the MCA at the receiving end of the message channel.

On z/OS, both the user ID received from the network, and the user ID derived from MCAUSER might be used, depending on the number of user IDs that are to be checked.

The queues are opened with this user ID and the open option MQOO_SET_ALL_CONTEXT.

Context security (CTX)
The user ID from the context information associated with the message is used as an alternate user ID.

The *UserIdentifier* in the message descriptor is moved into the *AlternateUserId* field in the object descriptor. The queue is opened with the open options MQOO_SET_ALL_CONTEXT and MQOO_ALTERNATE_USER_AUTHORITY.

On platforms other than z/OS, the user ID used to check open authority on the queue for MQOO_SET_ALL_CONTEXT and MQOO_ALTERNATE_USER_AUTHORITY is that of the process or user running the MCA at the receiving end of the message channel. The user ID used to check open authority on the queue for MQOO_OUTPUT is the *UserIdentifier* in the message descriptor.

On z/OS, the user ID received from the network or that derived from MCAUSER might be used, as well as the user ID from the context information in the message descriptor, depending on the number of user IDs that are to be checked.

Context security (CTX) is not supported on server-connection channels.

Only Message Channel Agent security (ONLYMCA)
The user ID derived from MCAUSER is used.

Queues are opened with the open option MQOO_SET_ALL_CONTEXT.

This value only applies to z/OS.

Alternate Message Channel Agent security (ALTMCA)
The user ID from the context information (the *UserIdentifier* field) in the message descriptor might be used, as well as the user ID derived from MCAUSER, depending on the number of user IDs that are to be checked.

This value only applies to z/OS.

Further details about context fields and open options can be found in Controlling context information.

More information about security can be found here:

- Securing

- ULW Setting up security on UNIX, Linux, and Windows
- IBM i Setting up security on IBM i
- z/OS Setting up security on z/OS

Queue manager name (QMNAME)

This attribute specifies the name of the queue manager or queue manager group to which an IBM MQ MQI client application can request connection.

This attribute is valid for channel types of:

- Client connection

Receive exit name (RCVEXIT)

This attribute specifies the name of the user exit program to be run by the channel receive user exit.

This attribute can be a list of names of programs that are to be run in succession. Leave blank, if no channel receive user exit is in effect.

The format and maximum length of this attribute depend on the platform:

- On z/OS it is a load module name, maximum length 8 characters, except for client-connection channels where the maximum length is 128 characters.
- On IBM i, it is of the form:

  ```
  libname/progname
  ```

  when specified in CL commands.

  When specified in IBM MQ Commands (MQSC) it has the form:

  ```
  progname libname
  ```

  where *progname* occupies the first 10 characters, and *libname* the second 10 characters (both blank-padded to the right if necessary). The maximum length of the string is 20 characters.
- On Windows, it is of the form:

  ```
  dllname(functionname)
  ```

  where *dllname* is specified without the suffix `.DLL`. The maximum length of the string is 40 characters.
- On UNIX, it is of the form:

  ```
  libraryname(functionname)
  ```

  The maximum length of the string is 40 characters.

During cluster sender channel auto-definition on z/OS, channel exit names are converted to z/OS format. If you want to control how exit names are converted, you can write a channel auto-definition exit. For more information, see Channel auto-definition exit program.

You can specify a list of receive, send, or message exit program names. The names must be separated by a comma, a space, or both. For example:

```
RCVEXIT(exit1 exit2)
MSGEXIT(exit1,exit2)
SENDEXIT(exit1, exit2)
```

The total length of the string of exit names and strings of user data for a particular type of exit is limited to 500 characters. In IBM MQ for IBM i, you can list up to 10 exit names. In IBM MQ for z/OS, you can list up to eight exit names.

This attribute is valid for all channel types.

Receive exit user data (RCVDATA)

This attribute specifies user data that is passed to the receive exit.

You can run a sequence of receive exits. The string of user data for a series of exits must be separated by a comma, spaces, or both. For example:

```
RCVDATA(exit1_data exit2_data)
MSGDATA(exit1_data,exit2_data)
SENDDATA(exit1_data, exit2_data)
```

In IBM MQ for UNIX systems, and Windows systems, the length of the string of exit names and strings of user data is limited to 500 characters. In IBM MQ for IBM i, you can specify up to 10 exit names and the length of user data for each is limited to 32 characters. In IBM MQ for z/OS, you can specify up to eight strings of user data each of length 32 characters.

This attribute is valid for all channel types.

Security exit name (SCYEXIT)

This attribute specifies the name of the exit program to be run by the channel security exit.

Leave blank if no channel security exit is in effect.

The format and maximum length of the name depend on the platform, as for "Receive exit name (RCVEXIT)" on page 130. However, you can only specify one security exit.

This attribute is valid for all channel types.

Security exit user data (SCYDATA)

This attribute specifies user data that is passed to the security exit.

The maximum length is 32 characters.

This attribute is valid for all channel types.

Send exit name (SENDEXIT)

This attribute specifies the name of the exit program to be run by the channel send exit.

This attribute can be a list of names of programs that are to be run in sequence. Leave blank if no channel send exit is in effect.

The format and maximum length of this attribute depend on the platform, as for "Receive exit name (RCVEXIT)" on page 130.

This attribute is valid for all channel types.

Send exit user data (SENDDATA)

This attribute specifies user data that is passed to the send exit.

You can run a sequence of send exits. The limitations on the user data length and an example of how to specify SENDDATA for more than one exit, are as shown for RCVDATA. See "Receive exit user data (RCVDATA)" on page 131.

This attribute is valid for all channel types.

Sequence number wrap (SEQWRAP)

This attribute specifies the highest number the message sequence number reaches before it restarts at 1.

The value of the number must be high enough to avoid a number being reissued while it is still being used by an earlier message. The two ends of a channel must have the same sequence number wrap value when a channel starts; otherwise, an error occurs.

The value can be set from 100 through 999 999 999.

This attribute is valid for channel types of:

- Sender
- Server
- Receiver
- Requester
- Cluster sender
- Cluster receiver

Short retry count (SHORTRTY)

This attribute specifies the maximum number of times that the channel is to try allocating a session to its partner.

The *short retry count* attribute can be set from zero through 999 999 999.

This attribute is valid for the following channel types:

- Sender
- Server
- Cluster sender
- Cluster receiver

If multiple IP addresses have been defined within the channel and reconnection is necessary, IBM MQ evaluates the channel definition and attempts to connect to each IP address in the order it is defined until either a successful connection is established or all addresses have been attempted.

In this case, SHORTRTY relates to how many total attempts the overall channel tries to reconnect, and not the individual IP addresses

If the initial allocation attempt fails, the *short retry count* is decremented and the channel retries the remaining number of times with an interval, defined in the **short retry interval** attribute, between each attempt. If it still fails, it retries *long retry count* number of times with an interval of *long retry interval* between each attempt. If it is still unsuccessful, the channel closes down.

On z/OS, a channel cannot enter retry if the maximum number of channels (**MAXCHL**) has been exceeded.

On IBM i, UNIX, and Windows systems, in order for retry to be attempted a channel initiator must be running. The channel initiator must be monitoring the initiation queue specified in the definition of the transmission queue that the channel is using.

If the channel initiator (on z/OS) or the channel (on Multiplatforms) is stopped while the channel is retrying, the *short retry count* and *long retry count* are reset when the channel initiator or the channel is restarted, or when a message is successfully put at the sender channel. However, if the channel initiator (on z/OS) or queue manager (on Multiplatforms) is shut down and restarted, the *short retry count* and *long retry count* are not reset. The channel retains the retry count values it had before the queue manager restart or the message being put.

For IBM i, UNIX, and Windows systems:

1. When a channel goes from RETRYING state to RUNNING state, the *short retry count* and *long retry count* are not reset immediately. They are reset only when the first message flows across the channel

successfully after the channel went into RUNNING state, that is; when the local channel confirms the number of messages sent to the other end.

2. The *short retry count* and *long retry count* are reset when the channel is restarted.

Short retry interval (SHORTTMR)

This attribute specifies the approximate interval in seconds that the channel is to wait before retrying to establish connection, during the short retry mode.

The interval between retries might be extended if the channel has to wait to become active.

This attribute can be set from zero through 999 999.

This attribute is valid for channel types of:

- Sender
- Server
- Cluster sender
- Cluster receiver

If multiple IP addresses have been defined within the channel and reconnection is necessary, IBM MQ evaluates the channel definition and attempts to connect to each IP address in the order it is defined until either a successful connection is established or all addresses have been attempted.

In this case, SHORTTMR relates to how long the overall channel waits to restart the connection process, and not the individual IP addresses.

SSL Cipher Specification (SSLCIPH)

This attribute specifies a single CipherSpec for a TLS connection.

Every IBM MQ channel definition includes the SSLCIPH attribute. The value is a string with a maximum length of 32 characters.

Note the following:

- The SSLCIPH attribute can contain a blank value, meaning that you are not using TLS. If one end of the channel has a blank SSLCIPH attribute, the other end of the channel must also have a blank SSLCIPH attribute.
- Alternatively, if SSLCIPH contains a nonblank value, the channel attempts to use the specified cipher to use TLS. Again, in this case, both ends of the channel must specify the same SSLCIPH value.
- The only exception to the rule that SSLCIPH must be the same at both ends of a channel, is that a fully-managed .NET client can specify the special value ***NEGOTIATE**. This option allows the channel to select the most recent protocol version supported by the .NET framework, and negotiate a CipherSpec that the server supports.

It is valid only for channels with a transport type (TRPTYPE) of TCP. If the TRPTYPE is not TCP, the data is ignored and no error message is issued.

For more information about SSLCIPH, see DEFINE CHANNEL and Specifying CipherSpecs.

SSL Client Authentication (SSLCAUTH)

This attribute specifies whether the channel needs to receive and authenticate a TLS certificate from a TLS client.

Possible values are:

OPTIONAL
If the peer TLS client sends a certificate, the certificate is processed as normal but authentication does not fail if no certificate is sent.

REQUIRED
If the TLS client does not send a certificate, authentication fails.

The default value is REQUIRED.

You can specify a value for SSLCAUTH on a non-TLS channel definition. That is, a channel definition on which the SSLCIPH attribute is missing or blank.

SSLCAUTH is an optional attribute.

This attribute is valid on all channel types that can ever receive a channel initiation flow, except for sender channels.

This attribute is valid for channel types of:

- Server
- Receiver
- Requester
- Server connection
- Cluster receiver

For more information about SSLCAUTH, see DEFINE CHANNEL (MQTT) and Securing.

SSL Peer (SSLPEER)

This attribute is used to check the Distinguished Name (DN) of the certificate from the peer queue manager or client at the other end of an IBM MQ channel.

Note: An alternative way of restricting connections into channels by matching against the TLS Subject Distinguished Name, is to use channel authentication records. With channel authentication records, different TLS Subject Distinguished Name patterns can be applied to the same channel. If both SSLPEER on the channel and a channel authentication record are used to apply to the same channel, the inbound certificate must match both patterns in order to connect.

If the DN received from the peer does not match the SSLPEER value, the channel does not start.

SSLPEER is an optional attribute. If a value is not specified, the peer DN is not checked when the channel is started.

On z/OS, the maximum length of the attribute is 256 bytes. On all other platforms, it is 1024 bytes. Channel authentication records provide greater flexibility when using SSLPEER and support 1024 bytes on all platforms.

On z/OS, the attribute values used are not checked. If you enter incorrect values, the channel fails at startup, and error messages are written to the error log at both ends of the channel. A Channel SSL Error event is also generated at both ends of the channel. On platforms that support SSLPEER, other than z/OS, the validity of the string is checked when it is first entered.

You can specify a value for SSLPEER on a non-TLS channel definition, one on which SSLCIPH is missing or blank. You can use this to temporarily disable TLS for debugging without having to clear and later re-input the TLS parameters.

For more information about using SSLPEER, see SET CHLAUTH and Securing.

This attribute is valid for all channel types.

Related concepts

Channel authentication records

V 9.0.0 Topic root (TPROOT)

This attribute specifies the topic root for an AMQP channel.

You can use the TPROOT attribute to specify a topic root for an AMQP channel. Using this attribute ensures that an MQ Light application, when deployed to a queue manager, does not publish or subscribe to messages to or from areas of the topic tree that are being used by other applications.

The default value for TPROOT is SYSTEM.BASE.TOPIC. With this value, the topic string an AMQP client uses to publish or subscribe has no prefix, and the client can exchange messages with other MQ pub/sub applications. To have AMQP clients publish and subscribe under a topic prefix, first create an MQ topic object with a topic string set to the prefix you want, then change the value of the AMQP channel TPROOT attribute to the name of the MQ topic object you created. The following example shows the topic root being set to APPGROUP1.BASE.TOPIC for AMQP channel MYAMQP:

```
DEFINE CHANNEL(MYAMQP) CHLTYPE(AMQP) TPROOT(APPGROUP1.BASE.TOPIC) PORT(5673)
```

Note: If the TPROOT attribute value, or the topic string that underpins it, is changed, existing AMQP topics and their messages might be orphaned.

Transmission queue name (XMITQ)

This attribute specifies the name of the transmission queue from which messages are retrieved.

This attribute is required for channels of type sender or server, it is not valid for other channel types.

Provide the name of the transmission queue to be associated with this sender or server channel, that corresponds to the queue manager at the far side of the channel. You can give the transmission queue the same name as the queue manager at the remote end.

This attribute is valid for channel types of:

- Sender
- Server

Transport type (TRPTYPE)

This attribute specifies the transport type to be used.

The possible values are:

LU62	LU 6.2
TCP	TCP/IP
NETBIOS	NetBIOS (“1” on page 135)
SPX	SPX (“1” on page 135)
Notes: 1. For use on Windows. Can also be used on z/OS for defining client-connection channels for use on Windows.	

This attribute is valid for all channel types, but is ignored by responding message channel agents.

Use client ID (USECLTID)

Use client ID for connection to AMQP channel.

Specify whether the client ID is used for connection on an AMQP channel. Set to Yes or No.

Use Dead-Letter Queue (USEDLQ)

This attribute determines whether the dead-letter queue (or undelivered message queue) is used when messages cannot be delivered by channels.

Possible values are:

NO
: Messages that cannot be delivered by a channel are treated as a failure. The channel either discards these messages, or the channel ends, in accordance with the setting of NPMSPEED.

YES (default)
If the queue manager DEADQ attribute provides the name of a dead-letter queue, then it is used, otherwise the behavior is as for NO.

User ID (USERID)

This attribute specifies the user ID to be used by the MCA when attempting to initiate a secure SNA session with a remote MCA.

You can specify a task user identifier of 20 characters.

It is valid for channel types of sender, server, requester, or client-connection.

This attribute does not apply to IBM MQ for z/OS except for client-connection channels.

On the receiving end, if passwords are kept in encrypted format and the LU 6.2 software is using a different encryption method, an attempt to start the channel fails with invalid security details. You can avoid this failure by modifying the receiving SNA configuration to either:

- Turn off password substitution, or
- Define a security user ID and password.

On IBM MQ for z/OS, this attribute is valid only for client connection channels. On other platforms, it is valid for channel types of:

- Sender
- Server
- Requester
- Client connection
- Cluster sender

IBM MQ cluster commands

The IBM MQ Script commands **runmqsc** commands have special attributes and parameters that apply to clusters. There are other administrative interfaces you can use to manager clusters.

The MQSC commands are shown as they would be entered by the system administrator at the command console. Remember that you do not have to issue the commands in this way. There are a number of other methods, depending on your platform; for example:

- On IBM MQ for IBM i, you run MQSC commands interactively from option 26 of **WRKMQM**. You can also use CL commands or you can store MQSC commands in a file and use the **STRMQMMQSC** CL command.
- z/OS On z/OS you can use the COMMAND function of the **CSQUTIL** utility, the operations and control panels or you can use the z/OS console.
- On all other platforms, you can store the commands in a file and use **runmqsc**.

In a MQSC command, a cluster name, specified using the CLUSTER attribute, can be up to 48 characters long.

A list of cluster names, specified using the CLUSNL attribute, can contain up to 256 names. To create a cluster namelist, use the DEFINE NAMELIST command.

IBM MQ Explorer

The IBM MQ Explorer GUI can administer a cluster with repository queue managers on IBM WebSphere MQ for z/OS 6 or later. You do not need to nominate an additional repository on a separate system. For earlier versions of IBM MQ for z/OS, the IBM MQ Explorer cannot administer a cluster with repository queue managers. You must therefore nominate an additional repository on a system that the IBM MQ Explorer can administer.

On IBM MQ for Windows and IBM MQ for Linux, you can also use IBM MQ Explorer to work with clusters. You can also use the stand-alone IBM MQ Explorer client.

Using the IBM MQ Explorer, you can view cluster queues and inquire about the status of cluster-sender and cluster-receiver channels. IBM MQ Explorer includes two wizards, which you can use to guide you through the following tasks:

- Create a cluster
- Join an independent queue manager to a cluster

Programmable command formats (PCF)

Table 31. PCF equivalents of MQSC commands specifically to work with clusters

runmqsc command	PCF equivalent
DISPLAY CLUSQMGR	MQCMD_INQUIRE_CLUSTER_Q_MGR
SUSPEND QMGR	MQCMD_SUSPEND_Q_MGR_CLUSTER
RESUME QMGR	MQCMD_RESUME_Q_MGR_CLUSTER
REFRESH CLUSTER	MQCMD_REFRESH_CLUSTER
RESET CLUSTER	MQCMD_RESET_CLUSTER

Related information

Clustering: Using REFRESH CLUSTER best practices

Queue manager definition commands

Cluster attributes that can be specified on queue manager definition commands.

To specify that a queue manager holds a full repository for a cluster, use the ALTER QMGR command specifying the attribute REPOS(*clustername*). To specify a list of several cluster names, define a cluster namelist and then use the attribute REPOSNL(*namelist*) on the ALTER QMGR command:

```
DEFINE NAMELIST(CLUSTERLIST)
       DESCR('List of clusters whose repositories I host')
       NAMES(CLUS1, CLUS2, CLUS3)
 ALTER QMGR REPOSNL(CLUSTERLIST)
```

You can provide additional cluster attributes on the ALTER QMGR command

CLWLEXIT(*name*)
Specifies the name of a user exit to be called when a message is put to a cluster queue.

CLWLDATA(*data*)
Specifies the data to be passed to the cluster workload user exit.

CLWLLEN(*length*)
Specifies the maximum amount of message data to be passed to the cluster workload user exit

CLWLMRUC(*channels*)
Specifies the maximum number of outbound cluster channels.

CLWLMRUC is a local queue manager attribute that is not propagated around the cluster. It is made available to cluster workload exits and the cluster workload algorithm that chooses the destination for messages.

CLWLUSEQ(LOCAL|ANY)
Specifies the behavior of MQPUT when the target queue has both a local instance and at least one remote cluster instance. If the put originates from a cluster channel, this attribute does not apply. It is possible to specify CLWLUSEQ as both a queue attribute and a queue manager attribute.

If you specify ANY, both the local queue and the remote queues are possible targets of the MQPUT.

If you specify LOCAL, the local queue is the only target of the MQPUT.

The equivalent PCFs are MQCMD_CHANGE_Q_MGR and MQCMD_INQUIRE_Q_MGR.

Related concepts

Workload balancing in clusters
If a cluster contains more than one instance of the same queue, IBM MQ selects a queue manager to route a message to. It uses the cluster workload management algorithm, and a number of cluster workload-specific attributes, to determine the best queue manager to use.

z/OS Asynchronous behavior of CLUSTER commands on z/OS
The command issuer of a cluster command on z/OS receives confirmation a command has been sent, but not that it has completed successfully.

Related reference

Channel definition commands
Cluster attributes that can be specified on channel definition commands.

Queue definition commands
Cluster attributes that can be specified on the queue definition commands.

DISPLAY CLUSQMGR
Use the DISPLAY CLUSQMGR command to display cluster information about queue managers in a cluster.

SUSPEND QMGR, RESUME QMGR and clusters
Use the SUSPEND QMGR and RESUME QMGR command to temporarily reduce the inbound cluster activity to this queue manager, for example, before you perform maintenance on this queue manager, and then reinstate it.

REFRESH CLUSTER
Issue the REFRESH CLUSTER command from a queue manager to discard all locally held information about a cluster. You are unlikely to need to use this command, except in exceptional circumstances.

RESET CLUSTER: Forcibly removing a queue manager from a cluster
Use the **RESET CLUSTER** command to forcibly remove a queue manager from a cluster in exceptional circumstances.

Channel definition commands

Cluster attributes that can be specified on channel definition commands.

The DEFINE CHANNEL, ALTER CHANNEL, and DISPLAY CHANNEL commands have two specific CHLTYPE parameters for clusters: CLUSRCVR and CLUSSDR. To define a cluster-receiver channel you use the DEFINE CHANNEL command, specifying CHLTYPE(CLUSRCVR). Many attributes on a cluster-receiver channel definition are the same as the attributes on a receiver or sender-channel definition. To define a cluster-sender channel you use the DEFINE CHANNEL command, specifying CHLTYPE(CLUSSDR), and many of the same attributes as you use to define a sender-channel.

It is no longer necessary to specify the name of the full repository queue manager when you define a cluster-sender channel. If you know the naming convention used for channels in your cluster, you can make a CLUSSDR definition using the +QMNAME+ construction. The +QMNAME+ construction is not supported on z/OS. After connection, IBM MQ changes the name of the channel and substitutes the correct full repository queue manager name in place of +QMNAME+. The resulting channel name is truncated to 20 characters.

For more information on naming conventions, see Cluster naming conventions.

The technique works only if your convention for naming channels includes the name of the queue manager. For example, you define a full repository queue manager called QM1 in a cluster called CLUSTER1 with a cluster-receiver channel called CLUSTER1.QM1.ALPHA. Every other queue manager can define a cluster-sender channel to this queue manager using the channel name, CLUSTER1.+QMNAME+.ALPHA.

If you use the same naming convention for all your channels, be aware that only one +QMNAME+ definition can exist at one time.

The following attributes on the DEFINE CHANNEL and ALTER CHANNEL commands are specific to cluster channels:

CLUSTER
The CLUSTER attribute specifies the name of the cluster with which this channel is associated. Alternatively use the CLUSNL attribute.

CLUSNL
The CLUSNL attribute specifies a namelist of cluster names.

NETPRTY
Cluster-receivers only.

The NETPRTY attribute specifies a network priority for the channel. NETPRTY helps the workload management routines. If there is more than one possible route to a destination, the workload management routine selects the one with the highest priority.

CLWLPRTY
The CLWLPRTY parameter applies a priority factor to channels to the same destination for workload management purposes. This parameter specifies the priority of the channel for the purposes of cluster workload distribution. The value must be in the range zero through 9, where zero is the lowest priority and 9 is the highest.

CLWLRANK
The CLWLRANK parameter applies a ranking factor to a channel for workload management purposes. This parameter specifies the rank of a channel for the purposes of cluster workload distribution. The value must be in the range zero through 9, where zero is the lowest rank and 9 is the highest.

CLWLWGHT
The CLWLWGHT parameter applies a weighting factor to a channel for workload management purposes. CLWLWGHT weights the channel so that the proportion of messages sent down that channel can be controlled. The cluster workload algorithm uses CLWLWGHT to bias the destination choice so that more messages can be sent over a particular channel. By default all channel weight attributes are the same default value. The weight attribute allows you to allocate a channel on a powerful UNIX machine a larger weight than another channel on small desktop PC. The greater weight means that the cluster workload algorithm selects the UNIX machine more frequently than the PC as the destination for messages.

CONNAME
The CONNAME specified on a cluster-receiver channel definition is used throughout the cluster to identify the network address of the queue manager. Take care to select a value for the CONNAME parameter that resolves throughout your IBM MQ cluster. Do not use a generic name. Remember that the value specified on the cluster-receiver channel takes precedence over any value specified in a corresponding cluster-sender channel.

These attributes on the DEFINE CHANNEL command and ALTER CHANNEL command also apply to the DISPLAY CHANNEL command.

Note: Auto-defined cluster-sender channels take their attributes from the corresponding cluster-receiver channel definition on the receiving queue manager. Even if there is a manually defined cluster-sender channel, its attributes are automatically modified to ensure that they match the attributes on the corresponding cluster-receiver definition. Beware that you can, for example, define a CLUSRCVR without specifying a port number in the CONNAME parameter, while manually defining a CLUSSDR that does specify a port number. When the auto-defined CLUSSDR replaces the manually defined one, the port number (taken from the CLUSRCVR) becomes blank. The default port number would be used and the channel would fail.

Note: The DISPLAY CHANNEL command does not display auto-defined channels. However, you can use the DISPLAY CLUSQMGR command to examine the attributes of auto-defined cluster-sender channels.

Use the DISPLAY CHSTATUS command to display the status of a cluster-sender or cluster-receiver channel. This command gives the status of both manually defined channels and auto-defined channels.

The equivalent PCFs are MQCMD_CHANGE_CHANNEL, MQCMD_COPY_CHANNEL, MQCMD_CREATE_CHANNEL, and MQCMD_INQUIRE_CHANNEL.

Omitting the CONNAME value on a CLUSRCVR definition

In some circumstances you can omit the CONNAME value on a CLUSRCVR definition. You must not omit the CONNAME value on z/OS.

Multi On Multiplatforms, the TCP/IP connection name parameter of a cluster-receiver channel is optional. If you leave the connection name blank, IBM MQ generates a connection name for you, assuming the default port and using the current IP address of the system. You can override the default port number, but still use the current IP address of the system. For each connection name leave the IP name blank, and provide the port number in parentheses; for example:

```
(1415)
```

The generated **CONNAME** is always in the dotted decimal (IPv4) or hexadecimal (IPv6) form, rather than in the form of an alphanumeric DNS host name.

This facility is useful when you have machines using Dynamic Host Configuration Protocol (DHCP). If you do not supply a value for the CONNAME on a CLUSRCVR channel, you do not need to change the CLUSRCVR definition. DHCP allocates you a new IP address.

If you specify a blank for the CONNAME on the CLUSRCVR definition, IBM MQ generates a CONNAME from the IP address of the system. Only the generated CONNAME is stored in the repositories. Other queue managers in the cluster do not know that the CONNAME was originally blank.

If you issue the DISPLAY CLUSQMGR command you see the generated CONNAME. However, if you issue the DISPLAY CHANNEL command from the local queue manager, you see that the CONNAME is blank.

If the queue manager is stopped and restarted with a different IP address, because of DHCP, IBM MQ regenerates the CONNAME and updates the repositories accordingly.

Related concepts

Workload balancing in clusters
If a cluster contains more than one instance of the same queue, IBM MQ selects a queue manager to route a message to. It uses the cluster workload management algorithm, and a number of cluster workload-specific attributes, to determine the best queue manager to use.

z/OS Asynchronous behavior of CLUSTER commands on z/OS
The command issuer of a cluster command on z/OS receives confirmation a command has been sent, but not that it has completed successfully.

Related reference

Queue manager definition commands
Cluster attributes that can be specified on queue manager definition commands.

Queue definition commands
Cluster attributes that can be specified on the queue definition commands.

DISPLAY CLUSQMGR
Use the DISPLAY CLUSQMGR command to display cluster information about queue managers in a cluster.

SUSPEND QMGR, RESUME QMGR and clusters
Use the SUSPEND QMGR and RESUME QMGR command to temporarily reduce the inbound cluster activity to this queue manager, for example, before you perform maintenance on this queue manager, and then reinstate it.

REFRESH CLUSTER
Issue the REFRESH CLUSTER command from a queue manager to discard all locally held information about a cluster. You are unlikely to need to use this command, except in exceptional circumstances.

RESET CLUSTER: Forcibly removing a queue manager from a cluster

Use the **RESET CLUSTER** command to forcibly remove a queue manager from a cluster in exceptional circumstances.

Queue definition commands

Cluster attributes that can be specified on the queue definition commands.

The DEFINE QLOCAL, DEFINE QREMOTE, and DEFINE QALIAS commands

The cluster attributes on the DEFINE QLOCAL, DEFINE QREMOTE, and DEFINE QALIAS commands, and the three equivalent ALTER commands, are:

CLUSTER
Specifies the name of the cluster to which the queue belongs.

CLUSNL
Specifies a namelist of cluster names.

DEFBIND
Specifies the binding to be used when an application specifies MQOO_BIND_AS_Q_DEF on the MQOPEN call. The options for this attribute are:

- Specify DEFBIND(OPEN) to bind the queue handle to a specific instance of the cluster queue when the queue is opened. DEFBIND(OPEN) is the default for this attribute.
- Specify DEFBIND(NOTFIXED) so that the queue handle is not bound to any instance of the cluster queue.
- Specify DEFBIND(GROUP) to allow an application to request that a group of messages are all allocated to the same destination instance.

When multiple queues with the same name are advertised in a Queue Manager Cluster, applications can choose whether to send all messages from this application to a single instance (MQOO_BIND_ON_OPEN), to allow the workload management algorithm to select the most suitable destination on a per message basis (MQOO_BIND_NOT_FIXED), or allow an application to request that a 'group' of messages be all allocated to the same destination instance (MQOO_BIND_ON_GROUP). The workload balancing is re-driven between groups of messages (without requiring an MQCLOSE and MQOPEN of the queue).

When you specify DEFBIND on a queue definition, the queue is defined with one of the attributes, MQBND_BIND_ON_OPEN, MQBND_BIND_NOT_FIXED, or MQBND_BIND_ON_GROUP. Either MQBND_BIND_ON_OPEN or MQBND_BIND_ON_GROUP must be specified when using groups with clusters.

We recommend that you set the DEFBIND attribute to the same value on all instances of the same cluster queue. Because MQOO_BIND_ON_GROUP is new in IBM WebSphere MQ 7.1, it must not be used if any of the applications opening this queue are connecting to IBM WebSphere MQ 7.0.1 or earlier queue managers.

CLWLRANK
Applies a ranking factor to a queue for workload management purposes. CLWLRANK parameter is not supported on model queues. The cluster workload algorithm selects a destination queue with the highest rank. By default CLWLRANK for all queues is set to zero.

If the final destination is a queue manager on a different cluster, you can set the rank of any intermediate gateway queue managers at the intersection of neighboring clusters. With the intermediate queue managers ranked, the cluster workload algorithm correctly selects a destination queue manager nearer the final destination.

The same logic applies to alias queues. The rank selection is made before the channel status is checked, and therefore even non-accessible queue managers are available for selection. This has the effect of allowing a message to be routed through a network, rather than having it select between two possible destinations (as the priority would). So, if a channel is not started to the place where the rank has indicated, the message is not routed to the next highest rank, but waits until a channel is available to that destination (the message is held on the transmit queue).

CLWLPRTY

Applies a priority factor to a queue for workload management purposes. The cluster workload algorithm selects a destination queue with the highest priority. By default priority for all queues is set to zero.

If there are two possible destination queues, you can use this attribute to make one destination failover to the other destination. The priority selection is made after the channel status is checked. All messages are sent to the queue with the highest priority unless the status of the channel to that destination is not as favorable as the status of channels to other destinations. This means that only the most accessible destinations are available for selection. This has the effect of prioritizing between multiple destinations that are all available.

CLWLUSEQ

Specifies the behavior of an MQPUT operation for a queue. This parameter specifies the behavior of an MQPUT operation when the target queue has a local instance and at least one remote cluster instance (except where the MQPUT originates from a cluster channel). This parameter is only valid for local queues.

Possible values are: QMGR (the behavior is as specified by the CLWLUSEQ parameter of the queue manager definition), ANY (the queue manager treats the local queue as another instance of the cluster queue, for the purposes of workload distribution), LOCAL (the local queue is the only target of the MQPUT operation, providing the local queue is put enabled). The MQPUT behavior depends upon the cluster workload management algorithm.

The DISPLAY QUEUE and DISPLAY QCLUSTER commands

The attributes on the DEFINE QLOCAL, DEFINE QREMOTE, and DEFINE QALIAS commands also apply to the DISPLAY QUEUE command.

To display information about cluster queues, specify a queue type of QCLUSTER or the keyword CLUSINFO on the DISPLAY QUEUE command, or use the command DISPLAY QCLUSTER.

The DISPLAY QUEUE or DISPLAY QCLUSTER command returns the name of the queue manager that hosts the queue (or the names of all queue managers if there is more than one instance of the queue). It also returns the system name for each queue manager that hosts the queue, the queue type represented, and the date and time at which the definition became available to the local queue manager. This information is returned using the CLUSQMGR, QMID, CLUSQT, CLUSDATE, and CLUSTIME attributes.

The system name for the queue manager (QMID), is a unique, system-generated name for the queue manager.

You can define a cluster queue that is also a shared queue. For example. on z/OS you can define:

```
DEFINE QLOCAL(MYQUEUE) CLUSTER(MYCLUSTER) QSGDISP(SHARED) CFSTRUCT(STRUCTURE)
```

The equivalent PCFs are MQCMD_CHANGE_Q, MQCMD_COPY_Q, MQCMD_CREATE_Q, and MQCMD_INQUIRE_Q.

Related concepts

Workload balancing in clusters
If a cluster contains more than one instance of the same queue, IBM MQ selects a queue manager to route a message to. It uses the cluster workload management algorithm, and a number of cluster workload-specific attributes, to determine the best queue manager to use.

z/OS Asynchronous behavior of CLUSTER commands on z/OS
The command issuer of a cluster command on z/OS receives confirmation a command has been sent, but not that it has completed successfully.

Related reference

Queue manager definition commands
Cluster attributes that can be specified on queue manager definition commands.

Channel definition commands

Cluster attributes that can be specified on channel definition commands.

DISPLAY CLUSQMGR
Use the `DISPLAY CLUSQMGR` command to display cluster information about queue managers in a cluster.

SUSPEND QMGR, RESUME QMGR and clusters
Use the `SUSPEND QMGR` and `RESUME QMGR` command to temporarily reduce the inbound cluster activity to this queue manager, for example, before you perform maintenance on this queue manager, and then reinstate it.

REFRESH CLUSTER
Issue the `REFRESH CLUSTER` command from a queue manager to discard all locally held information about a cluster. You are unlikely to need to use this command, except in exceptional circumstances.

RESET CLUSTER: Forcibly removing a queue manager from a cluster
Use the **`RESET CLUSTER`** command to forcibly remove a queue manager from a cluster in exceptional circumstances.

DISPLAY CLUSQMGR

Use the `DISPLAY CLUSQMGR` command to display cluster information about queue managers in a cluster.

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.

The information includes most channel attributes that apply to cluster-sender and cluster-receiver channels. In addition, the following attributes can be displayed:

CHANNEL
The cluster-receiver channel name for the queue manager.

CLUSDATE
The date at which the definition became available to the local queue manager.

CLUSTER
What clusters the queue manager is in.

CLUSTIME
The time at which the definition became available to the local queue manager.

DEFTYPE
How the queue manager was defined. `DEFTYPE` can be one of the following values:

CLUSSDR
A cluster sender-channel has been administratively defined on the local queue manager but not yet recognized by the target queue manager. To be in this state the local queue manager has defined a manual cluster-sender channel but the receiving queue manager has not accepted the cluster information. This may be due to the channel never having been established due to availability or to an error in the cluster-sender configuration, for example a mismatch in the CLUSTER property between the sender and receiver definitions. This is a transitory condition or error state and should be investigated.

CLUSSDRA
This value represents an automatically discovered cluster queue manager, no cluster-sender channel is defined locally. This is the DEFTYPE for cluster queue managers for which the local queue manager has no local configuration but has been informed of. For example

- If the local queue manager is a full repository queue manager it should be the DEFTYPE value for all partial repository queue managers in the cluster.
- If the local queue manager is a partial repository, this could be the host of a cluster queue that is being used from this local queue manager or from a second full repository queue manager that this queue manager has been told to work with.

If the DEFTYPE value is CLUSSDRA and the local and remote queue managers are both full repositories for the named cluster, the configuration is not correct as a locally defined cluster-sender channel must be defined to convert this to a DEFTYPE of CLUSSDRB.

CLUSSDRB
A cluster sender-channel has been administratively defined on the local queue manager and accepted as a valid cluster channel by the target queue manager. This is the expected DEFTYPE of a partial repository queue manager's manually configured full repository queue manager. It should also be the DEFTYPE of any CLUSQMGR from one full repository to another full repository in the cluster. Manual cluster-sender channels should not be configured to partial repositories or from a partial repository queue manager to more than one full repository. If a DEFTYPE of CLUSSDRB is seen in either of these situations it should be investigated and corrected.

CLUSRCVR
Administratively defined as a cluster-receiver channel on the local queue manager. This represents the local queue manager in the cluster.

Note: To identify which CLUSQMGRs are full repository queue managers for the cluster, see the QMTYPE property.

For more information on defining cluster channels, see Cluster channels.

QMTYPE
Whether it holds a full repository or only a partial repository.

STATUS
The status of the cluster-sender channel for this queue manager.

SUSPEND
Whether the queue manager is suspended.

VERSION
The version of the IBM MQ installation that the cluster queue manager is associated with.

The version has the format VVRRMMFF:

- VV: Version
- RR: Release
- MM: Maintenance level
- FF: Fix level

XMITQ
The cluster transmission queue used by the queue manager.

See also the `DISPLAY QCLUSTER` command. This is briefly described in DISPLAY QUEUE and in the `DISPLAY QUEUE` and `DISPLAY QCLUSTER` commands section of “Queue definition commands” on page 141. For examples of using `DISPLAY QCLUSTER`, search the information set for "DISPLAY QCLUSTER" and "DIS QCLUSTER".

Related concepts

Workload balancing in clusters
If a cluster contains more than one instance of the same queue, IBM MQ selects a queue manager to route a message to. It uses the cluster workload management algorithm, and a number of cluster workload-specific attributes, to determine the best queue manager to use.

z/OS Asynchronous behavior of CLUSTER commands on z/OS
The command issuer of a cluster command on z/OS receives confirmation a command has been sent, but not that it has completed successfully.

Related reference

Queue manager definition commands
Cluster attributes that can be specified on queue manager definition commands.

Channel definition commands

Cluster attributes that can be specified on channel definition commands.

Queue definition commands
Cluster attributes that can be specified on the queue definition commands.

SUSPEND QMGR, RESUME QMGR and clusters
Use the SUSPEND QMGR and RESUME QMGR command to temporarily reduce the inbound cluster activity to this queue manager, for example, before you perform maintenance on this queue manager, and then reinstate it.

REFRESH CLUSTER
Issue the REFRESH CLUSTER command from a queue manager to discard all locally held information about a cluster. You are unlikely to need to use this command, except in exceptional circumstances.

RESET CLUSTER: Forcibly removing a queue manager from a cluster
Use the **RESET CLUSTER** command to forcibly remove a queue manager from a cluster in exceptional circumstances.

MQSC command **DISPLAY CLUSQMGR**

SUSPEND QMGR, RESUME QMGR and clusters

Use the SUSPEND QMGR and RESUME QMGR command to temporarily reduce the inbound cluster activity to this queue manager, for example, before you perform maintenance on this queue manager, and then reinstate it.

While a queue manager is suspended from a cluster, it does not receive messages on cluster queues that it hosts if there is an available queue of the same name on an alternative queue manager in the cluster. However, messages that are explicitly targeted at this queue manager, or where the target queue is only available on this queue manager, are still directed to this queue manager.

Receiving further inbound messages while the queue manager is suspended can be prevented by stopping the cluster receiver channels for this cluster. To stop the cluster receiver channels for a cluster, use the FORCE mode of the SUSPEND QMGR command.

Related concepts

Workload balancing in clusters
If a cluster contains more than one instance of the same queue, IBM MQ selects a queue manager to route a message to. It uses the cluster workload management algorithm, and a number of cluster workload-specific attributes, to determine the best queue manager to use.

z/OS Asynchronous behavior of CLUSTER commands on z/OS
The command issuer of a cluster command on z/OS receives confirmation a command has been sent, but not that it has completed successfully.

Related tasks

Maintaining a queue manager

Related reference

Queue manager definition commands
Cluster attributes that can be specified on queue manager definition commands.

Channel definition commands
Cluster attributes that can be specified on channel definition commands.

Queue definition commands
Cluster attributes that can be specified on the queue definition commands.

DISPLAY CLUSQMGR
Use the DISPLAY CLUSQMGR command to display cluster information about queue managers in a cluster.

REFRESH CLUSTER
Issue the REFRESH CLUSTER command from a queue manager to discard all locally held information about a cluster. You are unlikely to need to use this command, except in exceptional circumstances.

RESET CLUSTER: Forcibly removing a queue manager from a cluster

Use the **RESET CLUSTER** command to forcibly remove a queue manager from a cluster in exceptional circumstances.

SUSPEND QMGR
RESUME QMGR

REFRESH CLUSTER

Issue the REFRESH CLUSTER command from a queue manager to discard all locally held information about a cluster. You are unlikely to need to use this command, except in exceptional circumstances.

There are three forms of this command:

REFRESH CLUSTER(clustername) REPOS(NO)
: The default. The queue manager retains knowledge of all locally defined cluster queue manager and cluster queues and all cluster queue managers that are full repositories. In addition, if the queue manager is a full repository for the cluster it also retains knowledge of the other cluster queue managers in the cluster. Everything else is removed from the local copy of the repository and rebuilt from the other full repositories in the cluster. Cluster channels are not stopped if REPOS(NO) is used. A full repository uses its CLUSSDR channels to inform the rest of the cluster that it has completed its refresh.

REFRESH CLUSTER(clustername) REPOS(YES)
: In addition to the default behavior, objects representing full repository cluster queue managers are also refreshed. It is not valid to use this option if the queue manager is a full repository, if used the command will fail with an error AMQ9406/CSQX406E logged. If it is a full repository, you must first alter it so that it is not a full repository for the cluster in question. The full repository location is recovered from the manually defined CLUSSDR definitions. After refreshing with REPOS(YES) has been issued the queue manager can be altered so that it is once again a full repository, if required.

REFRESH CLUSTER(*)
: Refreshes the queue manager in all the clusters it is a member of. If used with REPOS(YES) REFRESH CLUSTER(*) has the additional effect of forcing the queue manager to restart its search for full repositories from the information in the local CLUSSDR definitions. The search takes place even if the CLUSSDR channel connects the queue manager to several clusters.

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.

Related concepts

Workload balancing in clusters
If a cluster contains more than one instance of the same queue, IBM MQ selects a queue manager to route a message to. It uses the cluster workload management algorithm, and a number of cluster workload-specific attributes, to determine the best queue manager to use.

z/OS Asynchronous behavior of CLUSTER commands on z/OS
The command issuer of a cluster command on z/OS receives confirmation a command has been sent, but not that it has completed successfully.

Related reference

Queue manager definition commands
Cluster attributes that can be specified on queue manager definition commands.

Channel definition commands
Cluster attributes that can be specified on channel definition commands.

Queue definition commands
Cluster attributes that can be specified on the queue definition commands.

DISPLAY CLUSQMGR

Use the DISPLAY CLUSQMGR command to display cluster information about queue managers in a cluster.

SUSPEND QMGR, RESUME QMGR and clusters
Use the SUSPEND QMGR and RESUME QMGR command to temporarily reduce the inbound cluster activity to this queue manager, for example, before you perform maintenance on this queue manager, and then reinstate it.

RESET CLUSTER: Forcibly removing a queue manager from a cluster
Use the **RESET CLUSTER** command to forcibly remove a queue manager from a cluster in exceptional circumstances.

Related information

Clustering: Using REFRESH CLUSTER best practices

RESET CLUSTER: Forcibly removing a queue manager from a cluster

Use the **RESET CLUSTER** command to forcibly remove a queue manager from a cluster in exceptional circumstances.

You are unlikely to need to use this command, except in exceptional circumstances.

You can issue the **RESET CLUSTER** command only from full repository queue managers. The command takes two forms, depending on whether you reference the queue manager by name or identifier.

1. ```
   RESET CLUSTER( clustername
   ) QMNAME( qmname ) ACTION(FORCEREMOVE) QUEUES(NO)
   ```
2. ```
   RESET CLUSTER( clustername
   ) QMID( qmid ) ACTION(FORCEREMOVE) QUEUES(NO)
   ```

You cannot specify both QMNAME and QMID. If you use QMNAME, and there is more than one queue manager in the cluster with that name, the command is not run. Use QMID instead of QMNAME to ensure the **RESET CLUSTER** command is run.

Specifying QUEUES(NO) on a **RESET CLUSTER** command is the default. Specifying QUEUES(YES) removes references to cluster queues owned by the queue manager from the cluster. The references are removed in addition to removing the queue manager from the cluster itself.

The references are removed even if the cluster queue manager is not visible in the cluster; perhaps because it was previously forcibly removed, without the QUEUES option.

You might use the **RESET CLUSTER** command if, for example, a queue manager has been deleted but still has cluster-receiver channels defined to the cluster. Instead of waiting for IBM MQ to remove these definitions (which it does automatically) you can issue the **RESET CLUSTER** command to tidy up sooner. All other queue managers in the cluster are then informed that the queue manager is no longer available.

If a queue manager is temporarily damaged, you might want to tell the other queue managers in the cluster before they try to send it messages. **RESET CLUSTER** removes the damaged queue manager. Later, when the damaged queue manager is working again, use the **REFRESH CLUSTER** command to reverse the effect of **RESET CLUSTER** and return the queue manager to the cluster.If the queue manager is in a publish/subscribe cluster, you then need to reinstate any required proxy subscriptions. See REFRESH CLUSTER considerations for publish/subscribe clusters.

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.

Using the **RESET CLUSTER** command is the only way to delete auto-defined cluster-sender channels.

Important: If the auto-defined channel to be removed is in-doubt, RESET CLUSTER does not immediately remove that channel. In this situation you need to issue a RESOLVE CHANNEL command, prior to the RESET CLUSTER command.

You are unlikely to need this command in normal circumstances. The IBM Support Center might advise you to issue the command to tidy up the cluster information held by cluster queue managers. Do not use this command as a short cut to removing a queue manager from a cluster. The correct way to remove a queue manager from a cluster is described in Removing a queue manager from a cluster.

Because repositories retain information for only 90 days, after that time a queue manager that was forcibly removed can reconnect to a cluster. It reconnects automatically, unless it has been deleted. If you want to prevent a queue manager from rejoining a cluster, you need to take appropriate security measures.

All cluster commands, except **DISPLAY CLUSQMGR**, work asynchronously. Commands that change object attributes involving clustering update the object and send a request to the repository processor. Commands for working with clusters are checked for syntax, and a request is sent to the repository processor.

The requests sent to the repository processor are processed asynchronously, along with cluster requests received from other members of the cluster. Processing might take a considerable time if they have to be propagated around the whole cluster to determine if they are successful or not.

Related concepts

Workload balancing in clusters
If a cluster contains more than one instance of the same queue, IBM MQ selects a queue manager to route a message to. It uses the cluster workload management algorithm, and a number of cluster workload-specific attributes, to determine the best queue manager to use.

z/OS Asynchronous behavior of CLUSTER commands on z/OS
The command issuer of a cluster command on z/OS receives confirmation a command has been sent, but not that it has completed successfully.

Related reference

Queue manager definition commands
Cluster attributes that can be specified on queue manager definition commands.

Channel definition commands
Cluster attributes that can be specified on channel definition commands.

Queue definition commands
Cluster attributes that can be specified on the queue definition commands.

DISPLAY CLUSQMGR
Use the DISPLAY CLUSQMGR command to display cluster information about queue managers in a cluster.

SUSPEND QMGR, RESUME QMGR and clusters
Use the SUSPEND QMGR and RESUME QMGR command to temporarily reduce the inbound cluster activity to this queue manager, for example, before you perform maintenance on this queue manager, and then reinstate it.

REFRESH CLUSTER
Issue the REFRESH CLUSTER command from a queue manager to discard all locally held information about a cluster. You are unlikely to need to use this command, except in exceptional circumstances.

RESET CLUSTER (reset a cluster)

Workload balancing in clusters

If a cluster contains more than one instance of the same queue, IBM MQ selects a queue manager to route a message to. It uses the cluster workload management algorithm, and a number of cluster workload-specific attributes, to determine the best queue manager to use.

Suitable destinations are chosen, by the cluster workload management algorithm, 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. These attributes are described in the subtopics.

Note: Specify the cluster workload channel attributes on the cluster-receiver channels at the target queue managers. Any balancing you specify on the matching cluster-sender channels is likely to be ignored. See Cluster channels.

After you configure the cluster workload-specific attributes, if the configuration does not behave as you expected, explore the details of how the algorithm chooses a queue manager. See "The cluster workload management algorithm" on page 153. If the results of this algorithm do not meet your needs, you can write a cluster workload user exit program, and use this exit to route messages to the queue of your choice in the cluster. See Writing and compiling cluster workload exits.

Related concepts

z/OS Asynchronous behavior of CLUSTER commands on z/OS
The command issuer of a cluster command on z/OS receives confirmation a command has been sent, but not that it has completed successfully.

Related reference

Queue manager definition commands
Cluster attributes that can be specified on queue manager definition commands.

Channel definition commands
Cluster attributes that can be specified on channel definition commands.

Queue definition commands
Cluster attributes that can be specified on the queue definition commands.

DISPLAY CLUSQMGR
Use the `DISPLAY CLUSQMGR` command to display cluster information about queue managers in a cluster.

SUSPEND QMGR, RESUME QMGR and clusters
Use the `SUSPEND QMGR` and `RESUME QMGR` command to temporarily reduce the inbound cluster activity to this queue manager, for example, before you perform maintenance on this queue manager, and then reinstate it.

REFRESH CLUSTER
Issue the `REFRESH CLUSTER` command from a queue manager to discard all locally held information about a cluster. You are unlikely to need to use this command, except in exceptional circumstances.

RESET CLUSTER: Forcibly removing a queue manager from a cluster
Use the **RESET CLUSTER** command to forcibly remove a queue manager from a cluster in exceptional circumstances.

Cluster workload balancing - channel attributes

An alphabetical list of the channel attributes used in cluster workload balancing.

CLWLPRTY (Cluster workload priority)

The CLWLPRTY channel attribute specifies the priority order for channels for cluster workload distribution. The value must be in the range 0-9, where 0 is the lowest priority and 9 is the highest.

Use the `CLWLPRTY` channel attribute to set a priority order for the available cluster destinations. IBM MQ selects the destinations with the highest priority before selecting destinations with the lowest cluster destination priority. If there are multiple destinations with the same priority, it selects the least recently used destination.

If there are two possible destinations, you can use this attribute to allow failover. Messages go to the queue manager with the highest priority channel. If it becomes unavailable then messages go to the next highest priority queue manager. Lower priority queue managers act as reserves.

IBM MQ checks channel status before prioritizing the channels. Only available queue managers are candidates for selection.

Notes:

- Specify this attribute on the cluster-receiver channel at the target queue manager. Any balancing you specify on the matching cluster-sender channel is likely to be ignored. See Cluster channels.
- 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.

CLWLRANK (Cluster workload rank)

The **CLWLRANK** channel attribute specifies the rank of channels for cluster workload distribution. The value must be in the range 0-9, where 0 is the lowest rank and 9 is the highest.

Use the **CLWLRANK** channel attribute if you want control over the final destination for messages sent to a queue manager in another cluster. Control the choice of final destination by setting the rank of the channels connecting a queue manager to the gateway queue managers at the intersection of the clusters.

When you set **CLWLRANK**, messages take a specified route through the interconnected clusters towards a higher ranked destination. For example, messages arrive at a gateway queue manager that can send them to either of two queue managers using channels ranked 1 and 2. They are automatically sent to the queue manager connected by a channel with the highest rank, in this case the channel to the queue manager ranked 2.

IBM MQ gets the rank of channels before checking channel status. Getting the rank before checking channel status means that even non-accessible channels are available for selection. It allows messages to be routed through the network even if the final destination is unavailable.

Notes:

- Specify this attribute on the cluster-receiver channel at the target queue manager. Any balancing you specify on the matching cluster-sender channel is likely to be ignored. See Cluster channels.
- If you also used the priority attribute **CLWLPRTY**, IBM MQ selects between available destinations. If a channel is not available to the destination with the highest rank, the message is held on the transmission queue. It is released when the channel becomes available. The message does not get sent to the next available destination in the rank order.

CLWLWGHT (Cluster workload weight)

The CLWLWGHT channel attribute specifies the weight applied to CLUSSDR and CLUSRCVR channels for cluster workload distribution. The value must be in the range 1-99, where 1 is the lowest weight and 99 is the highest.

Use CLWLWGHT to send servers with more processing power more messages. The higher the channel weight, the more messages are sent over that channel.

Notes:

- Specify this attribute on the cluster-receiver channel at the target queue manager. Any balancing you specify on the matching cluster-sender channel is likely to be ignored. See Cluster channels.
- When CLWLWGHT is modified from the default of 50 on any channel, workload balancing becomes dependent on the total number of times each channel was chosen for a message sent to any clustered queue. For more information, see “The cluster workload management algorithm” on page 153.

NETPRTY (Network-connection priority)

The NETPRTY channel attribute specifies the priority for a CLUSRCVR channel. The value must be in the range 0-9, where 0 is the lowest priority and 9 is the highest.

Use the NETPRTY attribute to make one network the primary network, and another network the backup network. Given a set of equally ranked channels, clustering chooses the path with the highest priority when multiple paths are available.

A typical example of using the NETPRTY channel attribute is to differentiate between networks that have different costs or speeds and connect the same destinations.

Note: Specify this attribute on the cluster-receiver channel at the target queue manager. Any balancing you specify on the matching cluster-sender channel is likely to be ignored. See Cluster channels.

Related concepts

The cluster workload management algorithm
The workload management algorithm uses workload balancing attributes and many rules to select the final destination for messages being put onto cluster queues.

Related reference

Cluster workload balancing - queue attributes
An alphabetical list of queue attributes used in cluster workload balancing.

Cluster workload balancing - queue manager attributes
An alphabetical list of queue manager attributes used in cluster workload balancing.

Cluster workload balancing - queue attributes

An alphabetical list of queue attributes used in cluster workload balancing.

CLWLPRTY

The **CLWLPRTY** queue attribute specifies the priority of local, remote, or alias queues for cluster workload distribution. The value must be in the range 0-9, where 0 is the lowest priority and 9 is the highest.

Use the **CLWLPRTY** queue attribute to set a preference for destination queues. IBM MQ selects the destinations with the highest priority before selecting destinations with the lowest cluster destination priority. If there are multiple destinations with the same priority, it selects the least recently used destination.

IBM MQ obtains the priority of queue managers after checking channel status. Only available queue managers are candidates for selection.

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.

If there are two possible destinations, you can use this attribute to allow failover. The highest priority queue manager receives requests, lower priority queue managers act as reserves. If the highest priority queue manager fails, then the next highest priority queue manager that is available, takes over.

CLWLRANK

The **CLWLRANK** queue attribute specifies the rank of a local, remote, or alias queue for cluster workload distribution. The value must be in the range 0-9, where 0 is the lowest rank and 9 is the highest.

Use the **CLWLRANK** queue attribute if you want control over the final destination for messages sent to a queue manager in another cluster. When you set **CLWLRANK**, messages take a specified route through the interconnected clusters towards a higher ranked destination.

For example, you might have defined two identically configured gateway queue managers to improve the availability of a gateway. Suppose you have defined cluster alias queues at the gateways for a local queue

defined in the cluster. If the local queue becomes unavailable, you intend the message to be held at one of the gateways pending the queue becoming available again. To hold the queue at a gateway, you must define the local queue with a higher rank than the cluster alias queues at the gateway.

If you define the local queue with the same rank as the queue aliases and the local queue is unavailable, the message travels between the gateways. On finding the local queue unavailable the first gateway queue manager routes the message to the other gateway. The other gateway tries to deliver the message to the target local queue again. If the local queue is still unavailable, it routes the message back to the first gateway. The message keeps being moved back and forth between the gateways until the target local queue became available again. By giving the local queue a higher rank, even if the queue is unavailable, the message is not rerouted to a destination of lower rank.

IBM MQ obtains the rank of queues before checking channel status. Obtaining the rank before checking channel status means that even non-accessible queues are available for selection. It allows messages to be routed through the network even if the final destination is unavailable.

If you used the priority attribute IBM MQ selects between available destinations. If a channel is not available to the destination with the highest rank, the message is held on the transmission queue. It is released when the channel becomes available. The message does not get sent to the next available destination in the rank order.

CLWLUSEQ

The **CLWLUSEQ** queue attribute specifies whether a local instance of a queue is given preference as a destination over other instances in a cluster.

The **CLWLUSEQ** queue attribute is valid only for local queues. It only applies if the message is put by an application, or a channel that is not a cluster channel.

LOCAL
: The local queue is the only target of MQPUT, providing the local queue is put enabled. MQPUT behavior depends upon the cluster workload management.

QMGR
: The behavior is as specified by the **CLWLUSEQ** queue manager attribute.

ANY
: MQPUT treats the local queue the same as any other instance of the queue in the cluster for workload distribution.

Related concepts

The cluster workload management algorithm
The workload management algorithm uses workload balancing attributes and many rules to select the final destination for messages being put onto cluster queues.

Related reference

Cluster workload balancing - channel attributes
An alphabetical list of the channel attributes used in cluster workload balancing.

Cluster workload balancing - queue manager attributes
An alphabetical list of queue manager attributes used in cluster workload balancing.

Cluster workload balancing - queue manager attributes

An alphabetical list of queue manager attributes used in cluster workload balancing.

CLWLMRUC

The **CLWLMRUC** queue manager attribute sets the number of most recently chosen channels. The cluster workload management algorithm uses **CLWLMRUC** to restrict the number of active outbound cluster channels. The value must be in the range 1 - 999 999 999.

The initial default value is 999 999 999.

CLWLUSEQ

The **CLWLUSEQ** queue manager attribute specifies whether a local instance of a queue is given preference as a destination over other instances of the queue in a cluster. The attribute applies if the **CLWLUSEQ** queue attribute is set to QMGR.

The **CLWLUSEQ** queue attribute is valid only for local queues. It only applies if the message is put by an application, or a channel that is not a cluster channel.

LOCAL
: The local queue is the only target of MQPUT. LOCAL is the default.

ANY
: MQPUT treats the local queue the same as any other instance of the queue in the cluster for workload distribution.

Related concepts

The cluster workload management algorithm
The workload management algorithm uses workload balancing attributes and many rules to select the final destination for messages being put onto cluster queues.

Related reference

Cluster workload balancing - channel attributes
An alphabetical list of the channel attributes used in cluster workload balancing.

Cluster workload balancing - queue attributes
An alphabetical list of queue attributes used in cluster workload balancing.

The cluster workload management algorithm

The workload management algorithm uses workload balancing attributes and many rules to select the final destination for messages being put onto cluster queues.

The workload management algorithm is exercised every time a choice of destination is required:

- It is used at the point a cluster queue is opened, by using the MQOO_BIND_ON_OPEN option.
- It is used each time a message is put to a cluster queue when it is opened with MQOO_BIND_NOT_FIXED.
- It is used each time a new message group is started when MQOO_BIND_ON_GROUP is used to open a cluster queue.
- For topic host routing, it is used each time a message is published to a clustered topic. If the local queue manager is not a host for this topic, the algorithm is used to choose a host queue manager to route the message through.

The following section describes the workload management algorithm used when determining the final destination for messages being put onto cluster queues. These rules are influenced by the settings applied to the following attributes for queues, queue managers, and channels:

Table 32. Attributes for cluster workload management

Queues	Queue managers	Channels
• CLWLPRTY [1] • CLWLRANK [1] • CLWLUSEQ [1] • PUT / PUB	• CLWLUSEQ [1] • CLWLMRUC	• CLWLPRTY • CLWLRANK • CLWLWGHT • NETPRTY

Initially, the queue manager builds a list of possible destinations from two procedures:

[1] This attribute applies only when choosing a clustered queue, not when choosing a topic.

- Matching the target `ObjectName` and `ObjectQmgrName` with queue manager alias definitions that are shared in the same clusters as the queue manager.
- Finding unique routes (that is, channels) to a queue manager that hosts a queue with the name `ObjectName` and is in one of the clusters that the queue manager is a member of.

The algorithm steps through the following rules to eliminate destinations from the list of possible destinations.

1. Remote instances of queues or topics or remote CLUSRCVR channels that do not share a cluster with the local queue manager are eliminated.
2. If a queue or topic name is specified, remote CLUSRCVR channels that are not in the same cluster as the queue or topic are eliminated.

   Note: All remaining queues, topics and channels at this stage are made available to the cluster workload exit, if it is configured.
3. All channels to queue managers or queue manager aliases that have a CLWLRANK less than the maximum rank of all remaining channels or queue manager aliases are eliminated.
4. All queues (not queue manager aliases) with a CLWLRANK less than the maximum rank of all remaining queues are eliminated.
5. If more than one instance of a queue, topic, or queue manager alias remains, and if any are pub put enabled, all those that are put disabled are eliminated.

   Note: If only put disabled instances remain then only inquire operations will succeed, all other operations will fail with MQRC_CLUSTER_PUT_INHIBITED.
6. When choosing a queue, if the resulting set of queues contains the local instance of the queue, the local instance is typically used. The local instance of the queue is used if one of the following conditions are true:
   - The use-queue attribute of the queue, CLWLUSEQ, is set to LOCAL.
   - Both the following statements are true:
     - The use-queue attribute of the queue, CLWLUSEQ, is set to QMGR.
     - The use-queue attribute of the queue manager, CLWLUSEQ, is set to LOCAL.
   - The message is received over a cluster channel rather than by being put by a local application.
   - For locally defined queues that are defined with CLWLUSEQ(ANY), or which inherit that same setting from the queue manager, the following points are true, within the wider set of conditions that apply:
     - The local queue is chosen, based on the `status` of the locally-defined CLUSRCVR channels in the same cluster as the queue. This status is compared to the status of the CLUSSDR channels that would take the message to remotely defined queues of the same name.

       For example, there is one CLUSRCVR in the same cluster as the queue. That CLUSRCVR has STOPPING status, whereas the other queues of the same name in the cluster have RUNNING or INACTIVE status. In this case the remote channels will be chosen, and the local CLUSSDR channels are not used.
     - The local queue is chosen based on the `number` of CLUSRCVR channels, in any comparison with CLUSSDR channels of the same status, that would take the message to remotely defined queues of the same name.

       For example, there are four CLUSRCVR channels in the same cluster as the queue, and one CLUSSDR channel. All the channels have the same status of either INACTIVE or RUNNING. Therefore, there are five channels to choose from, and two instances of the queue. Four-fifths (80 percent) of the messages go to the local queue.
7. If more than one queue manager remains, if any are not suspended then all those that are suspended are eliminated.
8. If more than one remote instance of a queue or topic remains, all channels that are inactive or running are included. The state constants are listed:

- MQCHS_INACTIVE
- MQCHS_RUNNING

9. If no remote instance of a queue or topic remains, all channels that are in binding, initializing, starting, or stopping state are included. The state constants are listed:
   - MQCHS_BINDING
   - MQCHS_INITIALIZING
   - MQCHS_STARTING
   - MQCHS_STOPPING
10. If no remote instance of a queue or topic remains, all channels that are being tried again are included. The state constant is listed:
    - MQCHS_RETRYING
11. If no remote instance of a queue or topic remains, all channels in requesting, paused, or stopped state are included. The state constants are listed:
    - MQCHS_REQUESTING
    - MQCHS_PAUSED
    - MQCHS_STOPPED
    - MQCHS_SWITCHING
12. If more than one remote instance of a queue or topic on any queue manager remains, channels with the highest NETPRTY value for each queue manager are chosen.
13. All remaining channels and queue manager aliases other than channels and aliases with the highest priority, CLWLPRTY, are eliminated. If any queue manager aliases remain, channels to the queue manager are kept.
14. If a queue is being chosen:
    - All queues other than queues with the highest priority, CLWLPRTY, are eliminated, and channels are kept.
15. The remaining channels are then reduced to no more than the maximum allowed number of most recently-used channels, CLWLMRUC, by eliminating the channels with the lowest values of MQWDR.DestSeqNumber.

    Note: Internal cluster control messages are sent using the same cluster workload algorithm where appropriate.

After the list of valid destinations has been calculated, messages are workload balanced across them, using the following logic:

- When more than one remote instance of a destination remains and all channels to that destination have CLWLWGHT set to the default setting of 50, the least recently used channel is chosen. This approximately equates to a round-robin style of workload balancing when multiple remote instances exist.
- When more than one remote instance of a destination remains and one or more of the channels to those queues has CLWLWGHT set to a non-default setting (even if they all have a matching non-default value), then routing becomes dependent on the relative weightings of each channel and the total number of times each channel has previously been chosen when sending messages.
- When observing the distribution of messages for a single clustered queue with multiple instances, this can appear to lead to an unbalanced distribution across a subset of queue instances. This is because it is the historic use of each cluster sender channel from this queue manager that is being balanced, not just the message traffic for that queue. If this behavior is not desirable, complete one of the following steps:
  - Set CLWLWGHT to 50 on all cluster receiver channels if even distribution is required.

  - Or, if certain queue instances need to be weighted differently from others, define those queues in a dedicated cluster, with defined dedicated cluster receiver channels. This action isolates the workload balancing of these queues from others in the cluster.
- The historic data that is used to balance the channels is reset if any cluster workload attributes of available cluster receiver channels are altered or the status of a cluster receiver channel becomes available. Modification to the workload attributes of manually defined cluster sender channels does not reset the historic data.
- When you are considering cluster workload exit logic, the chosen channel is the one with the lowest MQWDR.DestSeqFactor. Each time a channel is chosen, this value is increased by approximately 1000/CLWLWGHT. If there is more than one channel with the lowest value, one of the channels with the lowest MQWDR.DestSeqNumber value is chosen.

The distribution of user messages is not always exact because administration and maintenance of the cluster causes messages to flow across channels. The result is an uneven distribution of user messages that can take some time to stabilize. Because of the mixture of administration and user messages, place no reliance on the exact distribution of messages during workload balancing.

Related reference

Cluster workload balancing - channel attributes
An alphabetical list of the channel attributes used in cluster workload balancing.

Cluster workload balancing - queue attributes
An alphabetical list of queue attributes used in cluster workload balancing.

Cluster workload balancing - queue manager attributes
An alphabetical list of queue manager attributes used in cluster workload balancing.

Asynchronous behavior of CLUSTER commands on z/OS

The command issuer of a cluster command on z/OS receives confirmation a command has been sent, but not that it has completed successfully.

For both REFRESH CLUSTER and RESET CLUSTER, 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 been completed successfully.

Any errors are reported to the z/OS console on the system where the channel initiator is running, they are not sent to the command issuer.

The asynchronous behavior is in contrast to CHANNEL commands. A message indicating that a channel command has been accepted is issued immediately. At some later time, when the command has been completed, a message indicating either normal or abnormal completion is sent to the command issuer.

Related concepts

Workload balancing in clusters
If a cluster contains more than one instance of the same queue, IBM MQ selects a queue manager to route a message to. It uses the cluster workload management algorithm, and a number of cluster workload-specific attributes, to determine the best queue manager to use.

Related tasks

Checking that async commands for distributed networks have finished

Related reference

Queue manager definition commands
Cluster attributes that can be specified on queue manager definition commands.

Channel definition commands
Cluster attributes that can be specified on channel definition commands.

Queue definition commands
Cluster attributes that can be specified on the queue definition commands.

DISPLAY CLUSQMGR

Use the `DISPLAY CLUSQMGR` command to display cluster information about queue managers in a cluster.

SUSPEND QMGR, RESUME QMGR and clusters
Use the `SUSPEND QMGR` and `RESUME QMGR` command to temporarily reduce the inbound cluster activity to this queue manager, for example, before you perform maintenance on this queue manager, and then reinstate it.

REFRESH CLUSTER
Issue the `REFRESH CLUSTER` command from a queue manager to discard all locally held information about a cluster. You are unlikely to need to use this command, except in exceptional circumstances.

RESET CLUSTER: Forcibly removing a queue manager from a cluster
Use the **RESET CLUSTER** command to forcibly remove a queue manager from a cluster in exceptional circumstances.

Channel programs

This section looks at the different types of channel programs (MCAs) available for use at the channels.

The names of the MCAs are shown in the following tables.

Table 33. Channel programs for Windows, UNIX and Linux systems

Program name	Direction of connection	Communication
amqrmppa		Any
runmqlsr	Inbound	Any
amqcrs6a	Inbound	LU 6.2
amqcrsta	Inbound	TCP
runmqchl	Outbound	Any
runmqchi	Outbound	Any

runmqlsr (Run IBM MQ listener), runmqchl (Run IBM MQ channel), and runmqchi (Run IBM MQ channel initiator) are control commands that you can enter at the command line.

amqcrsta is invoked for TCP channels on UNIX and Linux systems using inetd, where no listener is started.

amqcrs6a is invoked as a transaction program when using LU6.2

IBM i Intercommunication jobs

The following jobs are associated with Intercommunication on IBM i. The names are contained in the following table.

Table 34. Job names

Job name	Description
AMQCLMAA	Non-threaded Listener
AMQCRSTA	Non-threaded Responder Job
AMQRMPPA	Channel Pool Job
RUNMQCHI	Channel Initiator
RUNMQCHL	Channel Job
RUNMQLSR	Threaded Listener

IBM i Channel states on IBM i

Channel states are displayed on the Work with Channels panel

Table 35. Channel states on IBM i

State name	Meaning
STARTING	Channel is ready to begin negotiation with target MCA
BINDING	Establishing a session and initial data exchange
REQUESTING	Requester channel initiating a connection
RUNNING	Transferring or ready to transfer
PAUSED	Waiting for message-retry interval
STOPPING	Establishing whether to retry or stop
RETRYING	Waiting until next retry attempt
STOPPED	Channel stopped because of an error or because an end-channel command is issued
INACTIVE	Channel ended processing normally or channel never started
*None	No state (for server-connection channels only)

ULW Message channel planning example for UNIX, Linux, and Windows

This section provides a detailed example of how to connect two queue managers together so that messages can be sent between them.

The example illustrates the preparations required to enable an application using queue manager QM1 to put messages on a queue at queue manager QM2. An application running on QM2 can retrieve these messages, and send responses to a reply queue on QM1.

The example illustrates the use of TCP/IP connections. The example assumes that channels are to be triggered to start when the first message arrives on the transmission queue they are servicing. You must start the channel initiator in order for triggering to work.

This example uses SYSTEM.CHANNEL.INITQ as the initiation queue. This queue is already defined by IBM MQ. You can use a different initiation queue, but you must define it yourself and specify the name of the queue when you start the channel initiator.

ULW What the example for UNIX, Linux, and Windows shows

The example shows the IBM MQ commands (MQSC) that you can use.

In all the examples, the MQSC commands are shown as they would appear in a file of commands, and as they would be typed at the command line. The two methods look identical, but, to issue a command at the command line, you must first 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.

An alternative method is to create a file containing these commands. Any errors in the commands are then easy to correct. If you called your file mqsc.in then to run it on queue manager QMNAME use:

```
runmqsc QMNAME < mqsc.in > mqsc.out
```

You could verify the commands in your file before running it using:

```
runmqsc -v QMNAME < mqsc.in > mqsc.out
```

For portability, you should restrict the line length of your commands to 72 characters. Use a concatenation character to continue over more than one line. On Windows use Ctrl-z to end the input at the command line. On UNIX and Linux systems use Ctrl-d. Alternatively, use the **end** command.

Figure 7 on page 159 shows the example scenario.

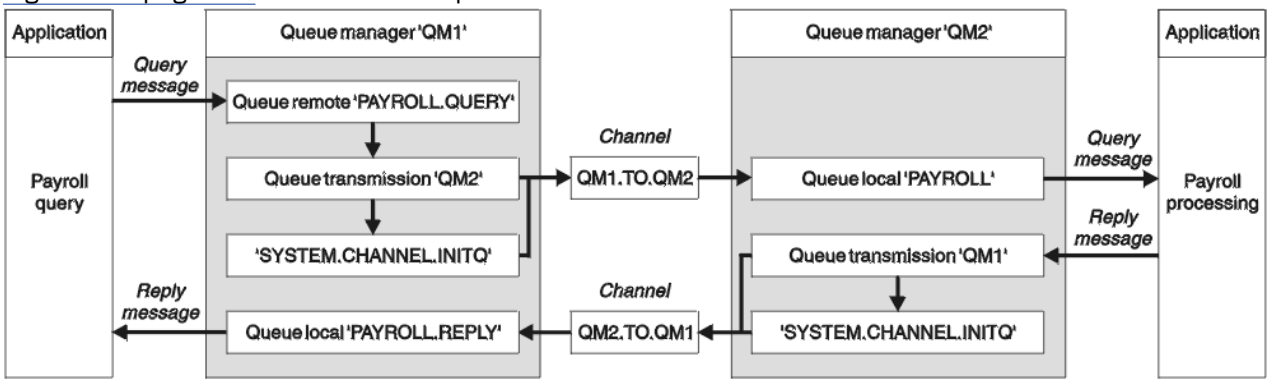

Figure 7. The message channel example for UNIX, Linux, and Windows systems

The example involves a payroll query application connected to queue manager QM1 that sends payroll query messages to a payroll processing application running on queue manager QM2. The payroll query application needs the replies to its queries sent back to QM1. The payroll query messages are sent from QM1 to QM2 on a sender-receiver channel called QM1.TO.QM2, and the reply messages are sent back from QM2 to QM1 on another sender-receiver channel called QM2.TO.QM1. Both of these channels are triggered to start as soon as they have a message to send to the other queue manager.

The payroll query application puts a query message to the remote queue "PAYROLL.QUERY" defined on QM1. This remote queue definition resolves to the local queue "PAYROLL" on QM2. In addition, the payroll query application specifies that the reply to the query is sent to the local queue "PAYROLL.REPLY" on QM1. The payroll processing application gets messages from the local queue "PAYROLL" on QM2, and sends the replies to wherever they are required; in this case, local queue "PAYROLL.REPLY" on QM1.

In the example definitions for TCP/IP, QM1 has a host address of 192.0.2.0 and is listening on port 1411, and QM2 has a host address of 192.0.2.1 and is listening on port 1412. The example assumes that these are already defined on your system and available for use.

The object definitions that need to be created on QM1 are:

- Remote queue definition, PAYROLL.QUERY
- Transmission queue definition, QM2 (default=remote queue manager name)
- Sender channel definition, QM1.TO.QM2
- Receiver channel definition, QM2.TO.QM1
- Reply-to queue definition, PAYROLL.REPLY

The object definitions that need to be created on QM2 are:

- Local queue definition, PAYROLL
- Transmission queue definition, QM1 (default=remote queue manager name)
- Sender channel definition, QM2.TO.QM1
- Receiver channel definition, QM1.TO.QM2

The connection details are supplied in the CONNAME attribute of the sender channel definitions.

You can see a diagram of the arrangement in Figure 7 on page 159.

ULW Queue manager QM1 example for UNIX, Linux, and Windows

These object definitions allow applications connected to queue manager QM1 to send request messages to a queue called PAYROLL on QM2, and to receive replies on a queue called PAYROLL.REPLY on QM1.

All the object definitions have been provided with the DESCR and REPLACE attributes. The other attributes supplied are the minimum required to make the example work. The attributes that are not supplied take the default values for queue manager QM1.

Run the following commands on queue manager QM1.

Remote queue definition

```
DEFINE QREMOTE(PAYROLL.QUERY) DESCR('Remote queue for QM2') REPLACE +
PUT(ENABLED) XMITQ(QM2) RNAME(PAYROLL) RQMNAME(QM2)
```

Note: The remote queue definition is not a physical queue, but a means of directing messages to the transmission queue, QM2, so that they can be sent to queue manager QM2.

Transmission queue definition

```
DEFINE QLOCAL(QM2) DESCR('Transmission queue to QM2') REPLACE +
USAGE(XMITQ) PUT(ENABLED) GET(ENABLED) TRIGGER TRIGTYPE(FIRST) +
INITQ(SYSTEM.CHANNEL.INITQ) PROCESS(QM1.TO.QM2.PROCESS)
```

When the first message is put on this transmission queue, a trigger message is sent to the initiation queue, SYSTEM.CHANNEL.INITQ. The channel initiator gets the message from the initiation queue and starts the channel identified in the named process.

Sender channel definition

```
DEFINE CHANNEL(QM1.TO.QM2) CHLTYPE(SDR) TRPTYPE(TCP) +
REPLACE DESCR('Sender channel to QM2') XMITQ(QM2) +
CONNAME('192.0.2.1(1412)')
```

Receiver channel definition

```
DEFINE CHANNEL(QM2.TO.QM1) CHLTYPE(RCVR) TRPTYPE(TCP) +
REPLACE DESCR('Receiver channel from QM2')
```

Reply-to queue definition

```
DEFINE QLOCAL(PAYROLL.REPLY) REPLACE PUT(ENABLED) GET(ENABLED) +
DESCR('Reply queue for replies to query messages sent to QM2')
```

The reply-to queue is defined as PUT(ENABLED). This ensures that reply messages can be put to the queue. If the replies cannot be put to the reply-to queue, they are sent to the dead-letter queue on QM1 or, if this queue is not available, remain on transmission queue QM1 on queue manager QM2. The queue has been defined as GET(ENABLED) to allow the reply messages to be retrieved.

ULW Queue manager QM2 example for UNIX, Linux, and Windows

The following object definitions allow applications connected to queue manager QM2 to retrieve request messages from a local queue called PAYROLL, and to put replies to these request messages to a queue called PAYROLL.REPLY on queue manager QM1.

You do not need to provide a remote queue definition to enable the replies to be returned to QM1. The message descriptor of the message retrieved from local queue PAYROLL contains both the reply-to queue and the reply-to queue manager names. Therefore, as long as QM2 can resolve the reply-to queue manager name to that of a transmission queue on queue manager QM2, the reply message can be sent. In this example, the reply-to queue manager name is QM1 and so queue manager QM2 requires a transmission queue of the same name.

All the object definitions have been provided with the DESCR and REPLACE attributes and are the minimum required to make the example work. The attributes that are not supplied take the default values for queue manager QM2.

Run the following commands on queue manager QM2.

Local queue definition

```
DEFINE QLOCAL(PAYROLL) REPLACE PUT(ENABLED) GET(ENABLED) +
DESCR('Local queue for QM1 payroll details')
```

This queue is defined as PUT(ENABLED) and GET(ENABLED) for the same reason as the reply-to queue definition on queue manager QM1.

Transmission queue definition

```
DEFINE QLOCAL(QM1) DESCR('Transmission queue to QM1') REPLACE +
USAGE(XMITQ) PUT(ENABLED) GET(ENABLED) TRIGGER TRIGTYPE(FIRST) +
INITQ(SYSTEM.CHANNEL.INITQ) PROCESS(QM2.TO.QM1.PROCESS)
```

When the first message is put on this transmission queue, a trigger message is sent to the initiation queue, SYSTEM.CHANNEL.INITQ. The channel initiator gets the message from the initiation queue and starts the channel identified in the named process.

Sender channel definition

```
DEFINE CHANNEL(QM2.TO.QM1) CHLTYPE(SDR) TRPTYPE(TCP) +
REPLACE DESCR('Sender channel to QM1') XMITQ(QM1) +
CONNAME('192.0.2.0(1411)')
```

Receiver channel definition

```
DEFINE CHANNEL(QM1.TO.QM2) CHLTYPE(RCVR) TRPTYPE(TCP) +
REPLACE DESCR('Receiver channel from QM1')
```

ULW Running the example for UNIX, Linux, and Windows

Information about starting the channel initiator and listener and suggestions for expanding on this scenario.

Once these definitions have been created, you need to:

- Start the channel initiator on each queue manager.
- Start the listener for each queue manager.

For information about starting the channel initiator and listener, see Setting up communication for Windows and Setting up communication on UNIX and Linux systems.

Expanding this example

This simple example could be expanded with:

- The use of LU 6.2 communications for interconnection with CICS systems, and transaction processing.
- Adding more queue, process, and channel definitions to allow other applications to send messages between the two queue managers.
- Adding user-exit programs on the channels to allow for link encryption, security checking, or additional message processing.
- Using queue manager aliases and reply-to queue aliases to understand more about how these can be used in the organization of your queue manager network.

IBM i Message channel planning example for IBM i

This section provides a detailed example of how to connect two IBM i queue managers together so that messages can be sent between them.

The example illustrates the preparations needed to allow an application using queue manager QM1 to put messages on a queue at queue manager QM2. An application running on QM2 can retrieve these messages, and send responses to a reply queue on QM1.

The example illustrates the use of TCP/IP connections. The example assumes that channels are to be triggered to start when the first message arrives on the transmission queue they are servicing.

This example uses SYSTEM.CHANNEL.INITQ as the initiation queue. This queue is already defined by IBM MQ. You can use a different initiation queue, but you have to define it yourself, start a new instance of the channel initiator using the STRMQMCHLI command, and provide it with the name of your initiation queue. For more information about triggering channels, see Triggering channels.

IBM i What the example for IBM i shows

This example involves a payroll query application connected to queue manager QM1 that sends payroll query messages to a payroll processing application running on queue manager QM2. The payroll query application needs the replies to its queries sent back to QM1.

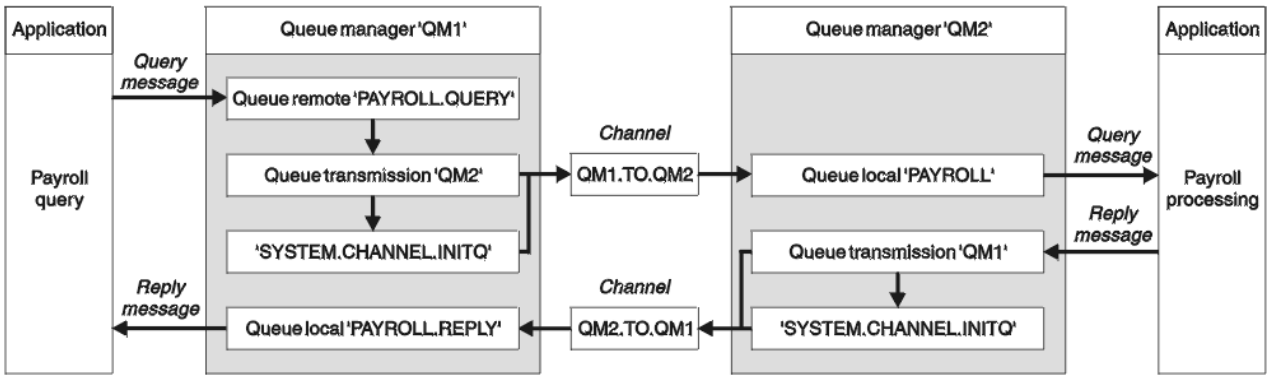

Figure 8. The message channel example for IBM MQ for IBM i

The payroll query messages are sent from QM1 to QM2 on a sender-receiver channel called QM1.TO.QM2, and the reply messages are sent back from QM2 to QM1 on another sender-receiver channel called QM2.TO.QM1. Both of these channels are triggered to start as soon as they have a message to send to the other queue manager.

The payroll query application puts a query message to the remote queue "PAYROLL.QUERY" defined on QM1. This remote queue definition resolves to the local queue "PAYROLL" on QM2. In addition, the payroll

query application specifies that the reply to the query is sent to the local queue "PAYROLL.REPLY" on QM1. The payroll processing application gets messages from the local queue "PAYROLL" on QM2, and sends the replies to wherever they are required; in this case, local queue "PAYROLL.REPLY" on QM1.

Both queue managers are assumed to be running on IBM i. In the example definitions, QM1 has a host address of 192.0.2.0 and is listening on port 1411. QM2 has a host address of 192.0.2.1 and is listening on port 1412. The example assumes that these queue managers are already defined on your IBM i system, and are available for use.

The object definitions that need to be created on QM1 are:

- Remote queue definition, PAYROLL.QUERY
- Transmission queue definition, QM2 (default=remote queue manager name)
- Sender channel definition, QM1.TO.QM2
- Receiver channel definition, QM2.TO.QM1
- Reply-to queue definition, PAYROLL.REPLY

The object definitions that need to be created on QM2 are:

- Local queue definition, PAYROLL
- Transmission queue definition, QM1 (default=remote queue manager name)
- Sender channel definition, QM2.TO.QM1
- Receiver channel definition, QM1.TO.QM2

The connection details are supplied in the CONNAME attribute of the sender channel definitions.

You can see a diagram of the arrangement in Figure 8 on page 162.

IBM i Queue manager QM1 example for IBM i

The following object definitions allow applications connected to queue manager QM1 to send request messages to a queue called PAYROLL on QM2, and to receive replies on a queue called PAYROLL.REPLY on QM1.

All the object definitions have been provided with the TEXT attributes. The other attributes supplied are the minimum required to make the example work. The attributes that are not supplied take the default values for queue manager QM1.

Run the following commands on queue manager QM1:

Remote queue definition

The CRTMQMQ command with the following attributes:

QNAME	'PAYROLL.QUERY'
QTYPE	*RMT
TEXT	'Remote queue for QM2'
PUTENBL	*YES
TMQNAME	'QM2' (default = remote queue manager name)
RMTQNAME	'PAYROLL'
RMTMQMNAME	'QM2'

Note: The remote queue definition is not a physical queue, but a means of directing messages to the transmission queue, QM2, so that they can be sent to queue manager QM2.

Transmission queue definition

The CRTMQMQ command with the following attributes:

QNAME	QM2

QTYPE	*LCL
TEXT	'Transmission queue to QM2'
USAGE	*TMQ
PUTENBL	*YES
GETENBL	*YES
TRGENBL	*YES
TRGTYPE	*FIRST
INITQNAME	SYSTEM.CHANNEL.INITQ
TRIGDATA	QM1.TO.QM2

When the first message is put on this transmission queue, a trigger message is sent to the initiation queue, SYSTEM.CHANNEL.INITQ. The channel initiator gets the message from the initiation queue and starts the channel identified in the named process.

Sender channel definition

The CRTMQMCHL command with the following attributes:

CHLNAME	QM1.TO.QM2
CHLTYPE	*SDR
TRPTYPE	*TCP
TEXT	'Sender channel to QM2'
TMQNAME	QM2
CONNAME	'192.0.2.1(1412)'

Receiver channel definition

The CRTMQMCHL command with the following attributes:

CHLNAME	QM2.TO.QM1
CHLTYPE	*RCVR
TRPTYPE	*TCP
TEXT	'Receiver channel from QM2'

Reply-to queue definition

The CRTMQMQ command with the following attributes:

QNAME	PAYROLL.REPLY
QTYPE	*LCL
TEXT	'Reply queue for replies to query messages sent to QM2'
PUTENBL	*YES
GETENBL	*YES

The reply-to queue is defined as PUT(ENABLED). This definition ensures that reply messages can be put to the queue. If the replies cannot be put to the reply-to queue, they are sent to the dead-letter queue on QM1 or, if this queue is not available, remain on transmission queue QM1 on queue manager QM2. The queue has been defined as GET(ENABLED) to allow the reply messages to be retrieved.

IBM i Queue manager QM2 example for IBM i

The following object definitions allow applications connected to queue manager QM2 to retrieve request messages from a local queue called PAYROLL, and to put replies to these request messages to a queue called PAYROLL.REPLY on queue manager QM1.

You do not need to provide a remote queue definition to enable the replies to be returned to QM1. The message descriptor of the message retrieved from local queue PAYROLL contains both the reply-to queue and the reply-to queue manager names. Therefore, as long as QM2 can resolve the reply-to queue manager name to that of a transmission queue on queue manager QM2, the reply message can be sent. In this example, the reply-to queue manager name is QM1 and so queue manager QM2 requires a transmission queue of the same name.

All the object definitions have been provided with the TEXT attribute and are the minimum required to make the example work. The attributes that are not supplied take the default values for queue manager QM2.

Run the following commands on queue manager QM2:

Local queue definition

The CRTMQMQ command with the following attributes:

QNAME	PAYROLL
QTYPE	*LCL
TEXT	'Local queue for QM1 payroll details'
PUTENBL	*YES
GETENBL	*YES

This queue is defined as PUT(ENABLED) and GET(ENABLED) for the same reason as the reply-to queue definition on queue manager QM1.

Transmission queue definition

The CRTMQMQ command with the following attributes:

QNAME	QM1
QTYPE	*LCL
TEXT	'Transmission queue to QM1'
USAGE	*TMQ
PUTENBL	*YES
GETENBL	*YES
TRGENBL	*YES
TRGTYPE	*FIRST
INITQNAME	SYSTEM.CHANNEL.INITQ
TRIGDATA	QM2.TO.QM1

When the first message is put on this transmission queue, a trigger message is sent to the initiation queue, SYSTEM.CHANNEL.INITQ. The channel initiator gets the message from the initiation queue and starts the channel identified in the trigger data.

Sender channel definition

The CRTMQMCHL command with the following attributes:

CHLNAME	QM2.TO.QM1
CHLTYPE	*SDR

TRPTYPE	*TCP
TEXT	'Sender channel to QM1'
TMQNAME	QM1
CONNAME	'192.0.2.0(1411)'

Receiver channel definition

The CRTMQMCHL command with the following attributes:

CHLNAME	QM1.TO.QM2
CHLTYPE	*RCVR
TRPTYPE	*TCP
TEXT	'Receiver channel from QM1'

IBM i Running the example for IBM i

When you have created the required objects you must start the channel initiators and listeners for both queue managers.

The applications can then send messages to each other. The channels are triggered to start by the first message arriving on each transmission queue, so you do not need to issue the STRMQMCHL command.

For details about starting a channel initiator and a listener, see Monitoring and controlling channels on IBM i .

IBM i Expanding the example for IBM i

The example can be expanded in a number of ways.

This example can be expanded by:

- Adding more queue and channel definitions to allow other applications to send messages between the two queue managers.
- Adding user exit programs on the channels to allow for link encryption, security checking, or additional message processing.
- Using queue manager aliases and reply-to queue aliases to understand more about how these objects can be used in the organization of your queue manager network.

For a version of this example that uses MQSC commands, see “Message channel planning example for z/OS” on page 166.

z/OS Message channel planning example for z/OS

This section provides a detailed example of how to connect z/OS or MVS queue managers together so that messages can be sent between them.

The example illustrates the preparations needed to allow an application using queue manager QM1 to put messages on a queue at queue manager QM2. An application running on QM2 can retrieve these messages, and send responses to a reply queue on QM1.

The example illustrates the use of both TCP/IP and LU 6.2 connections. The example assumes that channels are to be triggered to start when the first message arrives on the transmission queue they are servicing.

z/OS What the example for z/OS shows

This example involves a payroll query application connected to queue manager QM1 that sends payroll query messages to a payroll processing application running on queue manager QM2. The payroll query application needs the replies to its queries sent back to QM1.

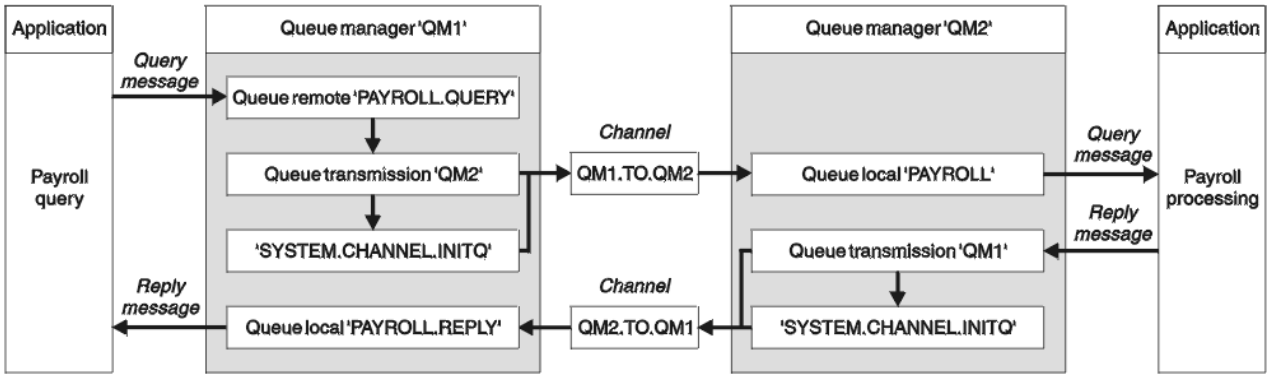

Figure 9. The first example for IBM MQ for z/OS

The payroll query messages are sent from QM1 to QM2 on a sender-receiver channel called QM1.TO.QM2, and the reply messages are sent back from QM2 to QM1 on another sender-receiver channel called QM2.TO.QM1. Both of these channels are triggered to start as soon as they have a message to send to the other queue manager.

The payroll query application puts a query message to the remote queue "PAYROLL.QUERY" defined on QM1. This remote queue definition resolves to the local queue "PAYROLL" on QM2. In addition, the payroll query application specifies that the reply to the query is sent to the local queue "PAYROLL.REPLY" on QM1. The payroll processing application gets messages from the local queue "PAYROLL" on QM2, and sends the replies to wherever they are required; in this case, local queue "PAYROLL.REPLY" on QM1.

Both queue managers are assumed to be running on z/OS. In the example definitions for TCP/IP, QM1 has a host address of 192.0.2.0 and is listening on port 1411, and QM2 has a host address of 192.0.2.1 and is listening on port 1412. In the definitions for LU 6.2, QM1 is listening on a symbolic luname called LUNAME1 and QM2 is listening on a symbolic luname called LUNAME2. The example assumes that these lunames are already defined on your z/OS system and available for use. To define them, see "Example MQ configuration for z/OS" on page 56.

The object definitions that need to be created on QM1 are:

- Remote queue definition, PAYROLL.QUERY
- Transmission queue definition, QM2 (default=remote queue manager name)
- Sender channel definition, QM1.TO.QM2
- Receiver channel definition, QM2.TO.QM1
- Reply-to queue definition, PAYROLL.REPLY

The object definitions that need to be created on QM2 are:

- Local queue definition, PAYROLL
- Transmission queue definition, QM1 (default=remote queue manager name)
- Sender channel definition, QM2.TO.QM1
- Receiver channel definition, QM1.TO.QM2

The example assumes that all the SYSTEM.COMMAND.* and SYSTEM.CHANNEL.* queues required to run DQM have been defined as shown in the supplied sample definitions, **CSQ4INSG** and **CSQ4INSX**.

The connection details are supplied in the CONNAME attribute of the sender channel definitions.

You can see a diagram of the arrangement in Figure 9 on page 167.

z/OS Queue manager QM1 example for z/OS

The following object definitions allow applications connected to queue manager QM1 to send request messages to a queue called PAYROLL on QM2. It also allows applications to receive replies on a queue called PAYROLL.REPLY on QM1.

All the object definitions have been provided with the DESCR and REPLACE attributes. The other attributes supplied are the minimum required to make the example work. The attributes that are not supplied take the default values for queue manager QM1.

Run the following commands on queue manager QM1.

Remote queue definition

```
DEFINE QREMOTE(PAYROLL.QUERY) DESCR('Remote queue for QM2') REPLACE +
PUT(ENABLED) XMITQ(QM2) RNAME(PAYROLL) RQMNAME(QM2)
```

Note: The remote queue definition is not a physical queue, but a means of directing messages to the transmission queue, QM2, so that they can be sent to queue manager QM2.

Transmission queue definition

```
DEFINE QLOCAL(QM2) DESCR('Transmission queue to QM2') REPLACE +
USAGE(XMITQ) PUT(ENABLED) GET(ENABLED) TRIGGER TRIGTYPE(FIRST) +
TRIGDATA(QM1.TO.QM2) INITQ(SYSTEM.CHANNEL.INITQ)
```

When the first message is put on this transmission queue, a trigger message is sent to the initiation queue, SYSTEM.CHANNEL.INITQ. The channel initiator gets the message from the initiation queue and starts the channel identified in the trigger data. The channel initiator can only get trigger messages from the SYSTEM.CHANNEL.INITQ queue, so do not use any other queue as the initiation queue.

Sender channel definition

For a TCP/IP connection:

```
DEFINE CHANNEL(QM1.TO.QM2) CHLTYPE(SDR) TRPTYPE(TCP) +
REPLACE DESCR('Sender channel to QM2') XMITQ(QM2) +
CONNAME('192.0.2.1(1412)')
```

For an LU 6.2 connection:

```
DEFINE CHANNEL(QM1.TO.QM2) CHLTYPE(SDR) TRPTYPE(LU62) +
REPLACE DESCR('Sender channel to QM2') XMITQ(QM2) +
CONNAME('LUNAME2')
```

Receiver channel definition

For a TCP/IP connection:

```
DEFINE CHANNEL(QM2.TO.QM1) CHLTYPE(RCVR) TRPTYPE(TCP) +
REPLACE DESCR('Receiver channel from QM2')
```

For an LU 6.2 connection:

```
DEFINE CHANNEL(QM2.TO.QM1) CHLTYPE(RCVR) TRPTYPE(LU62) +
REPLACE DESCR('Receiver channel from QM2')
```

Reply-to queue definition

```
DEFINE QLOCAL(PAYROLL.REPLY) REPLACE PUT(ENABLED) GET(ENABLED) +
DESCR('Reply queue for replies to query messages sent to QM2')
```

The reply-to queue is defined as PUT(ENABLED) which ensures that reply messages can be put to the queue. If the replies cannot be put to the reply-to queue, they are sent to the dead-letter queue on QM1 or, if this queue is not available, remain on transmission queue QM1 on queue manager QM2. The queue has been defined as GET(ENABLED) to allow the reply messages to be retrieved.

z/OS Queue manager QM2 example for z/OS

The following object definitions allow applications connected to queue manager QM2 to retrieve request messages from a local queue called PAYROLL and to put replies to these request messages to a queue called PAYROLL.REPLY on queue manager QM1.

You do not need to provide a remote queue definition to enable the replies to be returned to QM1. The message descriptor of the message retrieved from local queue PAYROLL contains both the reply-to queue and the reply-to queue manager names. Therefore, as long as QM2 can resolve the reply-to queue manager name to that of a transmission queue on queue manager QM2, the reply message can be sent. In this example, the reply-to queue manager name is QM1 and so queue manager QM2 requires a transmission queue of the same name.

All the object definitions have been provided with the DESCR and REPLACE attributes and are the minimum required to make the example work. The attributes that are not supplied take the default values for queue manager QM2.

Run the following commands on queue manager QM2.

Local queue definition

```
DEFINE QLOCAL(PAYROLL) REPLACE PUT(ENABLED) GET(ENABLED) +
DESCR('Local queue for QM1 payroll details')
```

This queue is defined as PUT(ENABLED) and GET(ENABLED) for the same reason as the reply-to queue definition on queue manager QM1.

Transmission queue definition

```
DEFINE QLOCAL(QM1) DESCR('Transmission queue to QM1') REPLACE +
USAGE(XMITQ) PUT(ENABLED) GET(ENABLED) TRIGGER TRIGTYPE(FIRST) +
TRIGDATA(QM2.TO.QM1) INITQ(SYSTEM.CHANNEL.INITQ)
```

When the first message is put on this transmission queue, a trigger message is sent to the initiation queue, SYSTEM.CHANNEL.INITQ. The channel initiator gets the message from the initiation queue and starts the channel identified in the trigger data. The channel initiator can only get trigger messages from SYSTEM.CHANNEL.INITQ so do not use any other queue as the initiation queue.

Sender channel definition

For a TCP/IP connection:

```
DEFINE CHANNEL(QM2.TO.QM1) CHLTYPE(SDR) TRPTYPE(TCP) +
REPLACE DESCR('Sender channel to QM1') XMITQ(QM1) +
CONNAME('192.0.2.0(1411)')
```

For an LU 6.2 connection:

```
DEFINE CHANNEL(QM2.TO.QM1) CHLTYPE(SDR) TRPTYPE(LU62) +
REPLACE DESCR('Sender channel to QM1') XMITQ(QM1) +
CONNAME('LUNAME1')
```

Receiver channel definition

For a TCP/IP connection:

```
DEFINE CHANNEL(QM1.TO.QM2) CHLTYPE(RCVR) TRPTYPE(TCP) +
REPLACE DESCR('Receiver channel from QM1')
```

For an LU 6.2 connection:

```
DEFINE CHANNEL(QM1.TO.QM2) CHLTYPE(RCVR) TRPTYPE(LU62) +
REPLACE DESCR('Receiver channel from QM1')
```

z/OS Running the example for z/OS

When you have created the required objects, you must start the channel initiators and listeners for both queue managers.

The applications can then send messages to each other. Because the channels are triggered to start by the arrival of the first message on each transmission queue, you do not need to issue the START CHANNEL MQSC command.

For details about starting a channel initiator see Starting a channel initiator, and for details about starting a listener see Starting a channel listener.

z/OS Expanding the example for z/OS

The example can be expanded in a number of ways.

The example can be expanded by:

- Adding more queue, and channel definitions to allow other applications to send messages between the two queue managers.
- Adding user exit programs on the channels to allow for link encryption, security checking, or additional message processing.
- Using queue manager aliases and reply-to queue aliases to understand more about how these aliases can be used in the organization of your queue manager network.

z/OS Message channel planning example for z/OS using queue sharing groups

This example illustrates the preparations needed to allow an application using queue manager QM3 to put a message on a queue in a queue sharing group that has queue members QM4 and QM5.

Ensure you are familiar with the example in “Message channel planning example for z/OS” on page 166 before trying this example.

z/OS What the queue sharing group example for z/OS shows

This example shows the IBM MQ commands (MQSC) that you can use in IBM MQ for z/OS for distributed queuing with queue sharing groups.

This example expands the payroll query scenario of the example in “Message channel planning example for z/OS” on page 166 to show how to add higher availability of query processing by adding more serving applications to serve a shared queue.

The payroll query application is now connected to queue manager QM3 and puts a query to the remote queue 'PAYROLL QUERY' defined on QM3. This remote queue definition resolves to the shared queue 'PAYROLL' hosted by the queue managers in the queue sharing group QSG1. The payroll processing application now has two instances running, one connected to QM4 and one connected to QM5.

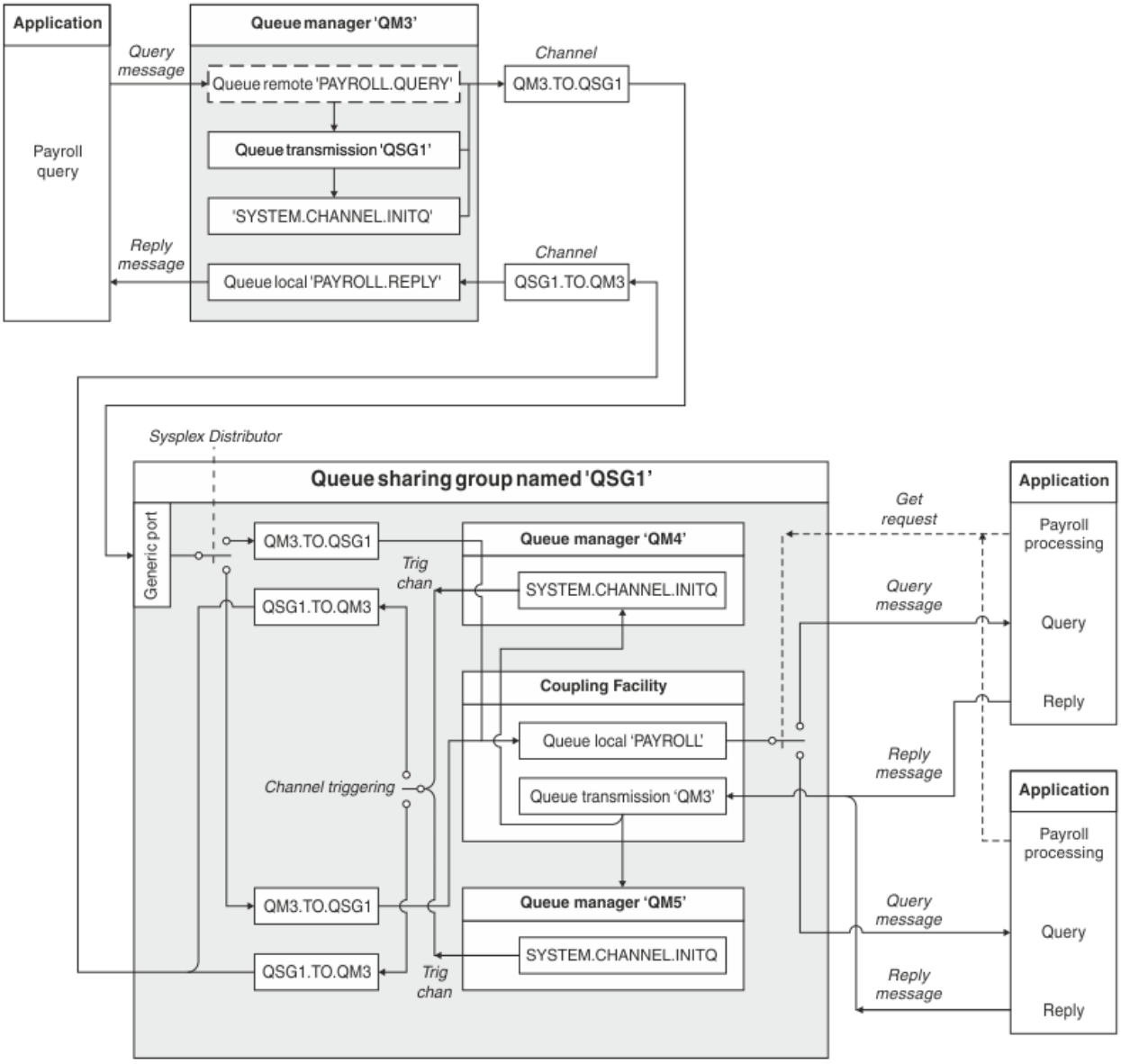

Figure 10. Message channel planning example for IBM MQ for z/OS using queue sharing groups

All three queue managers are assumed to be running on z/OS. In the example definitions for TCP/IP, QM4 has a VIPA address of MVSIP01 and QM5 has a VIPA address of MVSIP02. Both queue managers are listening on port 1414. The generic address that Sysplex Distributor provides for this group is QSG1.MVSIP. QM3 has a host address of 192.0.2.0 and is listening on port 1411.

In the example definitions for LU6.2, QM3 is listening on a symbolic luname called LUNAME1. The name of the generic resource defined for VTAM for the lunames listened on by QM4 and QM5 is LUQSG1. The example assumes that they are already defined on your z/OS system and are available for use. To define them see "Defining yourself to the network using generic resources" on page 63.

In this example QSG1 is the name of a queue sharing group, and queue managers QM4 and QM5 are the names of members of the group.

z/OS Queue sharing group definitions

Producing the following object definitions for one member of the queue sharing group makes them available to all the other members.

Queue managers QM4 and QM5 are members of the queue sharing group. The definitions produced for QM4 are also available for QM5.

It is assumed that the coupling facility list structure is called 'APPLICATION1'. If it is not called 'APPLICATION1', you must use your own coupling facility list structure name for the example.

Shared objects

The shared object definitions are stored in Db2® and their associated messages are stored within the coupling facility.

```
DEFINE QLOCAL(PAYROLL) QSGDISP(SHARED) REPLACE PUT(ENABLED) GET(ENABLED) +
CFSTRUCT(APPLICATION1) +
DESCR('Shared queue for payroll details')

DEFINE QLOCAL(QM3) QSGDISP(SHARED) REPLACE USAGE(XMITQ) PUT(ENABLED) +
CFSTRUCT(APPLICATION1) +
DESCR('Transmission queue to QM3') TRIGGER TRIGTYPE(FIRST) +
TRIGDATA(QSG1.TO.QM3) GET(ENABLED) INITQ(SYSTEM.CHANNEL.INITQ)
```

Group objects

The group object definitions are stored in Db2, and each queue manager in the queue sharing group creates a local copy of the defined object.

Sender channel definition for a TCP/IP connection:

```
DEFINE CHANNEL(QSG1.TO.QM3) CHLTYPE(SDR) QSGDISP(GROUP) TRPTYPE(TCP) +
REPLACE DESCR('Sender channel to QM3') XMITQ(QM3) +
CONNAME('192.0.2.0(1411)')
```

Sender channel definition for an LU 6.2 connection:

```
DEFINE CHANNEL(QSG1.TO.QM3) CHLTYPE(SDR) QSGDISP(GROUP) TRPTYPE(LU62) +
REPLACE DESCR('Sender channel to QM3') XMITQ(QM3) +
CONNAME('LUNAME1')
```

Receiver channel definition for a TCP/IP connection:

```
DEFINE CHANNEL(QM3.TO.QSG1) CHLTYPE(RCVR) TRPTYPE(TCP) +
REPLACE DESCR('Receiver channel from QM3') QSGDISP(GROUP)
```

Receiver channel definition for an LU 6.2 connection:

```
DEFINE CHANNEL(QM3.TO.QSG1) CHLTYPE(RCVR) TRPTYPE(LU62) +
REPLACE DESCR('Receiver channel from QM3') QSGDISP(GROUP)
```

Related reference

"Disposition (QSGDISP)" on page 116

This attribute specifies the disposition of the channel in a queue sharing group. It is valid on z/OS only.

z/OS Queue manager QM3 example for z/OS

QM3 is not a member of the queue sharing group. The following object definitions allow it to put messages to a queue in the queue sharing group.

The CONNAME for this channel is the generic address of the queue sharing group, which varies according to transport type.

For a TCP/IP connection:

```
DEFINE CHANNEL(QM3.TO.QSG1) CHLTYPE(SDR) TRPTYPE(TCP) +
REPLACE DESCR('Sender channel to QSG1') XMITQ(QSG1) +
CONNAME('QSG1.MVSIP(1414)')
```

For an LU 6.2 connection:

```
DEFINE CHANNEL(QM3.TO.QSG1) CHLTYPE(SDR) TRPTYPE(LU62) +
REPLACE DESCR('Sender channel to QSG1') XMITQ(QSG1) +
CONNAME('LUQSG1') TPNAME('MQSERIES') MODENAME('#INTER')
```

Other definitions

These definitions are required for the same purposes as the definitions in the first example.

```
DEFINE QREMOTE(PAYROLL.QUERY) DESCR('Remote queue for QSG1') REPLACE +
PUT(ENABLED) XMITQ(QSG1) RNAME(APPL) RQMNAME(QSG1)

DEFINE QLOCAL(QSG1) DESCR('Transmission queue to QSG1') REPLACE +
USAGE(XMITQ) PUT(ENABLED) GET(ENABLED) TRIGGER TRIGTYPE(FIRST) +
TRIGDATA(QM3.TO.QSG1) INITQ(SYSTEM.CHANNEL.INITQ)

DEFINE CHANNEL(QSG1.TO.QM3) CHLTYPE(RCVR) TRPTYPE(TCP) +
REPLACE DESCR('Receiver channel from QSG1')

DEFINE CHANNEL(QSG1.TO.QM3) CHLTYPE(RCVR) TRPTYPE(LU62) +
REPLACE DESCR('Receiver channel from QSG1')

DEFINE QLOCAL(PAYROLL.REPLY) REPLACE PUT(ENABLED) GET(ENABLED) +
DESCR('Reply queue for replies to query messages sent to QSG1')
```

z/OS Running the queue sharing group example for z/OS

When you have created the required objects you need to start the channel initiators for all three queue managers. You also need to start the listeners for both queue managers in the queue sharing group.

For a TCP/IP connection, each member of the group must have a group listener started that is listening on port 1414.

```
STA LSTR PORT(1414) IPADDR(MVSIP01) INDISP(GROUP)
```

The previous entry starts the listener on QM4, for example.

For an LU6.2 connection, each member of the group must have a group listener started that is listening on a symbolic luname. This luname must correspond to the generic resource LUQSG1.

- Start the listener on QM3

```
STA LSTR PORT(1411)
```

Using an alias to refer to an MQ library

You can define an alias to refer to an MQ library in your JCL, rather than use the name of the MQ library directly. Then, if the name of the MQ library changes, you have only to delete and redefine the alias.

Example

The following example defines an alias MQM.SCSQANLE to refer to the MQ library MQM.V600.SCSQANLE:

```
//STEP1 EXEC PGM=IDCAMS
//SYSPRINT DD SYSOUT=*
//SYSIN DD *
DELETE (MQM.SCSQANLE)
DEFINE ALIAS (NAME(MQM.SCSQANLE) RELATE(MQM.V600.SCSQANLE))
/*
```

Then, to refer to the MQM.V600.SCSQANLE library in your JCL, use the alias MQM.SCSQANLE.

Note: The library and alias names must be in the same catalog, so use the same high level qualifier for both; in this example, the high level qualifier is MQM.

z/OS V 9.0.1 `mqzOSConnectService` element

The MQ Service Provider is provided as a standard Liberty feature and so is configured using server.xml. Each one or two-way service is defined in an `mqzOSConnectService` element. This element and all of its attributes apply to both z/OS Connect V1 and z/OS Connect EE.

Important: An `mqzOSConnectService` element needs to be referenced by a `zOSConnectService` element before it can be used.

An example `mqzOSConnectService` element with some attributes specified is shown below.

```
<mqzOSConnectService id="twoWay "
                        connectionFactory="jms/cf1"
                        destination="jms/requestQueue"
                        replyDestination="jms/replyQueue"
                        expiry="-1"
                        waitInterval="10000"
                        replySelection="msgIDToCorrelID"
                        selector=""
                        persistence="false"/>
```

Attention: Depending on how the MQ Service Provider has been installed, the `mqzOSConnectService` element might be prefixed with a string followed by an underscore, for example `usr_mqzOSConnectService`.

This is described in Installing the MQ Service Provider into WLP for z/OS Connect V1 and Installing the MQ Service Provider into IBM z/OS Connect EE for z/OS Connect EE.

The format shown in the following example is where the MQ Service Provider has been installed into the WLP kernel (as described in option 1 of Installing the MQ Service Provider into WLP

Table 36. Attributes of an `mqzOSConnectService` element

Attribute name	Type	Default value	Description
id	string		“id” on page 175
connectionFactory	A JNDI name (string).		“connectionFactory” on page 175
destination	A JNDI name (string).		“destination” on page 175

Table 36. Attributes of an `mqzOSConnectService` element (continued)			
Attribute name	**Type**	**Default value**	**Description**
replyDestination	A JNDI name (string).		"replyDestination" on page 176
expiry	integer	-1	"expiry" on page 176
waitInterval	integer		"waitInterval" on page 176
replySelection	string	`msgIDToCorrelID`	"replySelection" on page 176
selector	string		"selector" on page 177
persistence	boolean	false	"persistence" on page 177
mqmdFormat	string		"mqmdFormat" on page 177
userName	string		"userName" on page 178
password	string		"password" on page 178
useCallerPrincipal	boolean	false	"useCallerPrincipal" on page 178
receiveTextCCSID	integer	37	"receiveTextCCSID" on page 178

id

`id` is a required attribute and must be unique across all elements in server.xml. **`id`** is used by the `zosConnectService` element to refer to a target service provider instance.

connectionFactory

`connectionFactory` specifies the JNDI name of an IBM MQ messaging provider connection factory. The MQ Service Provider uses the connection factory to connect to IBM MQ.

`connectionFactory` is a required attribute. For more information on connection factories, see JMS Connection Factory.

You should specify **`transportType`**="*BINDINGS*" for the connection factory.

destination

`destination` specifies the JNDI name of an IBM MQ messaging provider destination.

`destination` is a required attribute.

For more information on configuring a:

- Queue in WLP, see JMS Queue.
- Topic in WLP, see JMS Topic.

For a one-way service, **`destination`** is used as the target for HTTP POST, HTTP GET, and HTTP DELETE requests.

Note that queue destinations are supported for all three request types whereas topic destinations are supported only with HTTP POST requests.

For a two-way service, **`destination`** must be a queue destination which represents the request queue used by the back end service.

Two-way services support only HTTP POST requests.

replyDestination

replyDestination specifies the JNDI name of an IBM MQ messaging provider queue.

replyDestination is an optional attribute.

For more information on configuring a queue in WLP, see JMS Queue.

If **replyDestination** is not specified, the service is a one-way service. If **replyDestination** is specified, the service is a two-way service.

This queue is the reply destination where the back end service sends reply messages to.

expiry

expiry specifies how long messages sent by the MQ Service Provider are valid for, in thousandths of a second, from the time they were sent. The message becomes eligible to be discarded if it has not been removed from the destination queue before this period of time elapses.

expiry is an optional attribute, and is equivalent to setting the MQMD Expiry field.

Negative values means that messages never expire. The default value of **expiry** is -1.

REST clients can override **expiry** by specifying an `ibm-mq-md-expiry` HTTP header with a valid 64-bit integer.

waitInterval

For HTTP DELETE requests to one-way services, **waitInterval** specifies the number of milliseconds that the service waits for a matching message on the queue, specified by the **destination** attribute.

For HTTP POST requests to two-way services, **waitInterval** specifies the number of milliseconds that the service waits for a matching message on the queue, specified by the **replydestination** attribute.

waitInterval is an optional attribute for one-way services, a required attribute for two-way services, and is equivalent to setting the MQMD WaitInterval field.

waitInterval is not supported with HTTP GET requests.

If **waitInterval** is:

- Zero, the service does not wait.

  A **waitInterval** of zero is not supported with two-way services.
- Negative, the service waits for ever until a message is available.

REST clients can override this value by specifying an `ibm-mq-gmo-waitInterval` HTTP header with a valid 64 bit integer.

Note: Specifying a large, or negative **waitInterval**, is likely to result in transaction timeouts and asynchronous service request timeouts. If either, or both, of these events occur, increase the timeout, reduce the wait interval, or do both.

replySelection

`replySelection` describes the mechanism used to match reply messages with request messages.

`replySelection` is optional and used only used with two-way services. If `replySelection` is used with a one-way service, it is ignored.

The value is one of the following:

msgIDToCorrelID
Reply messages are assumed to be generated with the correlation ID set to the value of the message ID from the request message. The service generates a suitable message selector based on this information. This is the default value.

none
No mechanism is used to correlate reply messages with request messages. The service gets the first available message on the reply queue.

correlIDToCorrelID
Reply messages are assumed to be generated with the correlation ID set to the value of the correlation ID from the request message. The service generates a suitable message selector based on this information. If the request message does not have a correlation ID specified (see "ibm-mq-md-correlID" on page 179) the service generates a random correlation ID for the request message.

selector

`selector` must be a valid JMS message selector as described by the JMS specification.

`selector` is only used with one-way services and is optional. If `selector` is specified on a two-way service it is ignored. For more information on selectors, see Message selectors in JMS.

`selector` is used on HTTP GET and HTTP DELETE requests to select which message is returned. If the "ibm-mq-md-msgID" on page 178 or "ibm-mq-md-correlID" on page 179 headers are specified, `selector` is ignored.

Some selector characters need to be encoded in order to be embedded in `server.xml`. You can do this using standard mechanisms as follows:

```
" becomes &quot;
' becomes &apos;
< becomes &lt;
> becomes &gt;
```

persistence

`persistence` specifies the persistence of messages sent by a service.

`persistence` is optional, and is equivalent to setting the MQMD Persistence field.

The value is one of the following:

false
Means messages are non-persistent. This is the default value.

true
Means messages are persistent.

You can override `persistence` by using an `ibm-mq-md-persistence` HTTP header which takes the same values.

mqmdFormat

This attribute is used to set the value of the MQMD format field in messages that are sent by the MQ Service Provider. However, it is only used when the MQ Service Provider has been configured to use z/OS Connect data transformations, otherwise it is ignored.

If you do not specify this attribute, and data transformations are used, messages are sent with the MQMD format field set to blanks. The value of this attribute must be less than, or equal to, eight characters in length.

userName

The user name that the MQ Service Provider presents to IBM MQ for authentication and authorization purposes.

If you do not specify this attribute, the **`userName`** attribute in the connection factory referred to by the **`connectionFactory`** attribute is used.

If a **`userName`** attribute is specified, both on the referenced connection factory and on the MQ Service Provider, the MQ Service Provider value is used.

If you specify this attribute, you must specify the **`password`** attribute.

password

The password that the MQ Service Provider presents to IBM MQ for authentication and authorization purposes.

You can specify the password in plain text, although you should not do so. Instead, you should encode the password using the **`securityUtility`** tool provided with z/OS Connect, using the encode option. For more information see Liberty: securityUtility command.

If you do not specify this attribute, the password attribute in the connection factory referred to by the **`connectionFactory`** attribute is used.

If a password attribute is specified both on the referenced connection factory and on the MQ Service Provider the MQ Service Provider value is used.

If you specify this attribute, you must also specify the **`userName`** attribute.

useCallerPrincipal

When a request is made to z/OS Connect the caller authenticates with z/OS Connect. The name of the authenticated principle can be passed onto IBM MQ for authentication and authorization purposes.

To do this, set the value of **`useCallerPrincipal`** to true.

The name of the principal, but no password, is used when connecting to IBM MQ. Any values specified in the **`password`** and **`userName`** attributes are ignored.

receiveTextCCSID

The CCSID that is used when a data transformation is received and a `javax.jms.TextMessage` is being consumed (that is, an HTTP GET or HTTP DELETE with a one-way service, or on retrieving a response message for a two-way service).

The text in the message is converted into the CCSID specified by **`receiveTextCCSID`**.

z/OS V 9.0.1 HTTP headers that can be used with the MQ Service Provider

The only time the MQ Service Provider expects specific HTTP headers, is when an HTTP POST is issued.

In that case the Content-Type header must be set to "`application/json`". If you specify a character set as part of this header, its value must be utf-8.

For example `Content-Type=application/json;charset=utf-8`.

Other HTTP headers can be specified on the HTTP request to change the behavior of the MQ Service Provider; these are detailed in the following sections. Any other HTTP headers are ignored.

ibm-mq-md-msgID

This header can be specified when issuing HTTP GET or HTTP DELETE requests to one-way services.

The value of this header is used to generate a message selector to select a message with the specified message ID. If an “ibm-mq-md-correlID” on page 179 header is also specified, a message selector that matches both IDs will be generated.

See `msgId`: HTTP `x-msg-msgId` entity-header for details of the value of the format of this header.

ibm-mq-md-correlID

This header can be specified when issuing an HTTP POST, in which case it is used to set the MQMD CorrelID field of the message that gets sent.

This header can also be specified when issuing HTTP GET or DELETE requests to one-way services. The value of this header is used to generate a message selector to select a message with the specified correlation ID. If an “ibm-mq-md-msgID” on page 178 header is also specified, a message selector that matches both will be generated.

See `correlId`: HTTP `x-msg-correlId` entity-header for details of the value of the format of this header.

ibm-mq-pmo-retain

You can specify this header with a value of TRUE when issuing an HTTP POST request to a one-way service backed by a topic. This results in a retained publication being generated. For more information, see Retained publications.

ibm-mq-usr

You can use this header to provide message properties on the IBM MQ messages sent as a result of HTTP POST requests to both one-way and two-way services.

For details of the value of the format of this header, see usr: HTTP x-msg-usr entity-header.

Although the name used by the MQ Service Provider is different, see `require-headers`: HTTP `x-msg-require-headers` request-header for details of the value of the format of this header.

Notices

This information was developed for products and services offered in the U.S.A.

IBM may not offer the products, services, or features discussed in this document in other countries. Consult your local IBM representative for information on the products and services currently available in your area. Any reference to an IBM product, program, or service is not intended to state or imply that only that IBM product, program, or service may be used. Any functionally equivalent product, program, or service that does not infringe any IBM intellectual property right may be used instead. However, it is the user's responsibility to evaluate and verify the operation of any non-IBM product, program, or service.

IBM may have patents or pending patent applications covering subject matter described in this document. The furnishing of this document does not grant you any license to these patents. You can send license inquiries, in writing, to:

IBM Director of Licensing
IBM Corporation
North Castle Drive
Armonk, NY 10504-1785
U.S.A.

For license inquiries regarding double-byte (DBCS) information, contact the IBM Intellectual Property Department in your country or send inquiries, in writing, to:

Intellectual Property Licensing
Legal and Intellectual Property Law
IBM Japan, Ltd.
19-21, Nihonbashi-Hakozakicho, Chuo-ku
Tokyo 103-8510, Japan

The following paragraph does not apply to the United Kingdom or any other country where such provisions are inconsistent with local law: INTERNATIONAL BUSINESS MACHINES CORPORATION PROVIDES THIS PUBLICATION "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. Some states do not allow disclaimer of express or implied warranties in certain transactions, therefore, this statement may not apply to you.

This information could include technical inaccuracies or typographical errors. Changes are periodically made to the information herein; these changes will be incorporated in new editions of the publication. IBM may make improvements and/or changes in the product(s) and/or the program(s) described in this publication at any time without notice.

Any references in this information to non-IBM Web sites are provided for convenience only and do not in any manner serve as an endorsement of those Web sites. The materials at those Web sites are not part of the materials for this IBM product and use of those Web sites is at your own risk.

IBM may use or distribute any of the information you supply in any way it believes appropriate without incurring any obligation to you.

Licensees of this program who wish to have information about it for the purpose of enabling: (i) the exchange of information between independently created programs and other programs (including this one) and (ii) the mutual use of the information which has been exchanged, should contact:

IBM Corporation
Software Interoperability Coordinator, Department 49XA
3605 Highway 52 N
Rochester, MN 55901
U.S.A.

Such information may be available, subject to appropriate terms and conditions, including in some cases, payment of a fee.

The licensed program described in this information and all licensed material available for it are provided by IBM under terms of the IBM Customer Agreement, IBM International Program License Agreement, or any equivalent agreement between us.

Any performance data contained herein was determined in a controlled environment. Therefore, the results obtained in other operating environments may vary significantly. Some measurements may have been made on development-level systems and there is no guarantee that these measurements will be the same on generally available systems. Furthermore, some measurements may have been estimated through extrapolation. Actual results may vary. Users of this document should verify the applicable data for their specific environment.

Information concerning non-IBM products was obtained from the suppliers of those products, their published announcements or other publicly available sources. IBM has not tested those products and cannot confirm the accuracy of performance, compatibility or any other claims related to non-IBM products. Questions on the capabilities of non-IBM products should be addressed to the suppliers of those products.

All statements regarding IBM's future direction or intent are subject to change or withdrawal without notice, and represent goals and objectives only.

This information contains examples of data and reports used in daily business operations. To illustrate them as completely as possible, the examples include the names of individuals, companies, brands, and products. All of these names are fictitious and any similarity to the names and addresses used by an actual business enterprise is entirely coincidental.

COPYRIGHT LICENSE:

This information contains sample application programs in source language, which illustrate programming techniques on various operating platforms. You may copy, modify, and distribute these sample programs in any form without payment to IBM, for the purposes of developing, using, marketing or distributing application programs conforming to the application programming interface for the operating platform for which the sample programs are written. These examples have not been thoroughly tested under all conditions. IBM, therefore, cannot guarantee or imply reliability, serviceability, or function of these programs.

If you are viewing this information softcopy, the photographs and color illustrations may not appear.

Programming interface information

Programming interface information, if provided, is intended to help you create application software for use with this program.

This book contains information on intended programming interfaces that allow the customer to write programs to obtain the services of WebSphere MQ.

However, this information may also contain diagnosis, modification, and tuning information. Diagnosis, modification and tuning information is provided to help you debug your application software.

Important: Do not use this diagnosis, modification, and tuning information as a programming interface because it is subject to change.

Trademarks

IBM, the IBM logo, ibm.com®, are trademarks of IBM Corporation, registered in many jurisdictions worldwide. A current list of IBM trademarks is available on the Web at "Copyright and trademark information"www.ibm.com/legal/copytrade.shtml. Other product and service names might be trademarks of IBM or other companies.

Microsoft and Windows are trademarks of Microsoft Corporation in the United States, other countries, or both.

Part Number:

(1P) P/N: