7.5

IBM WebSphere MQ Configuration Reference

Note

Before using this information and the product it supports, read the information in “Notices” on page 141.

This edition applies to version 7 release 5 of IBM® WebSphere® MQ and to all subsequent releases and modifications until otherwise indicated in new editions.

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

Contents

Configuration reference

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

The configuration reference information is provided in the following subtopics:

Related tasks
Configuring

Example configuration information

The configuration examples describe tasks performed to establish a working WebSphere MQ network. The tasks are to establish WebSphere 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 in MQSC reference.

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

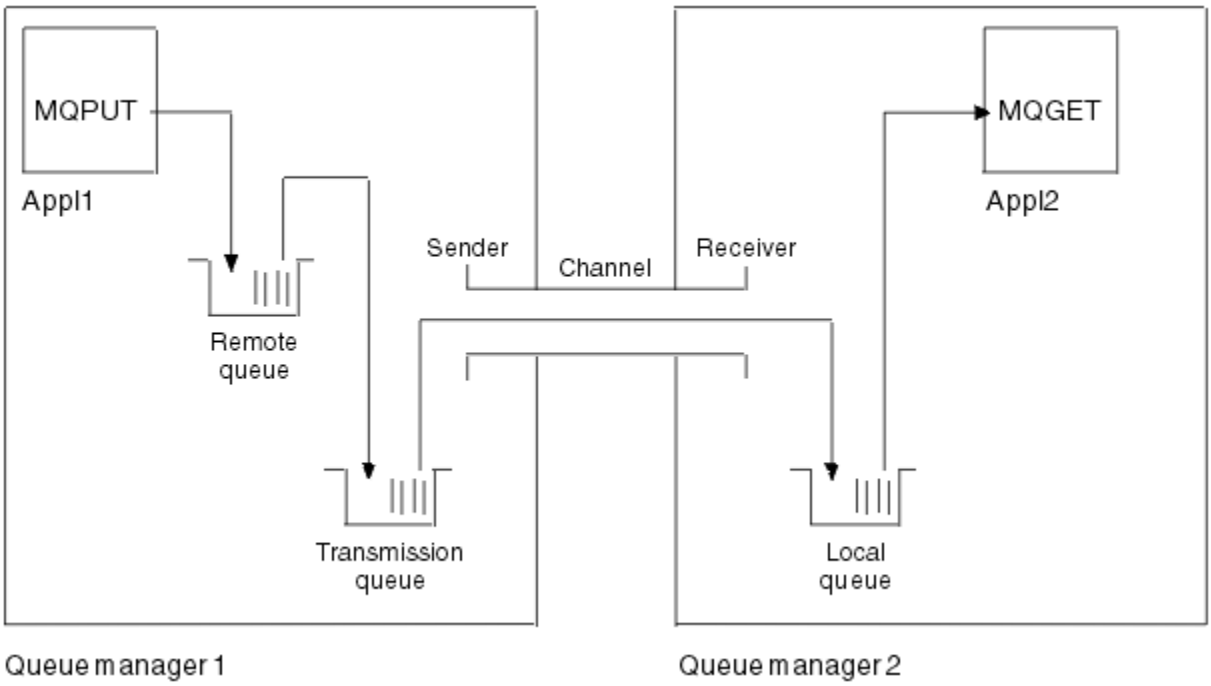

Figure 1. WebSphere MQ channel to be set up in the example configuration

This example is a simple one, intended to introduce only the basic elements of the WebSphere 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® communicates by using a 3745 network controller (or equivalent) that is attached to a token ring
- 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 V5.9
  - IBM Communications Server for AIX®, V6.3
  - Hewlett-Packard SNAplus2
  - IBM i
  - Data Connection SNAP-IX Version 7 or later
  - OS/390® Version 2 Release 4
- TCP
  - Microsoft Windows
  - AIX Version 4 Release 1.4
  - HP-UX Version 10.2 or later

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

Related tasks
Configuring

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.

If you are using a Windows, UNIX or Linux® system, before you begin to follow the instructions for your platform, you must set various environment variables. Set the environment variables by entering one of the following commands :

- On Windows:

  ```
  MQ_INSTALLATION_PATH/bin/setmqenv
  ```

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

- On UNIX and Linux systems:

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

  where *MQ_INSTALLATION_PATH* refers to the location where IBM WebSphere 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 worksheets in which you can find the parameters used in the example 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, record these values in the spaces on the worksheet. As you proceed through the section, you will find cross-references to these values as you need them.

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:

- “Example configuration - IBM WebSphere MQ for Windows” on page 8
- “Example configuration - IBM WebSphere MQ for AIX” on page 17
- “Example configuration - IBM WebSphere MQ for HP-UX” on page 23
- “Example configuration - IBM WebSphere MQ for Solaris” on page 29
- “Example configuration - IBM WebSphere MQ for Linux” on page 34

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

Related reference

setmqenv

Example configuration - IBM WebSphere MQ for Windows

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

Set up of communication links are 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 WebSphere MQ configuration" on page 11.

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

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

Establishing a TCP connection

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

The WebSphere MQ command used to start the WebSphere 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 WebSphere MQ configuration" on page 11.

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 WebSphere 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 WebSphere 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 WebSphere 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 WebSphere 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.

Establishing an SPX connection

An SPX connection applies only to a client and server running Windows XP and Windows 2003 Server.

This section contains information about:

- IPX/SPX parameters
- SPX addressing
- Receiving on SPX

IPX/SPX parameters

Refer to the Microsoft documentation for full details of the use and setting of the NWLink IPX and SPX parameters. The IPX/SPX parameters are in the following paths in the registry:

```
HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Service\NWLinkSPX\Parameters
HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Service\NWLinkIPX\Parameters
```

SPX addressing

WebSphere MQ uses the SPX address of each machine to establish connectivity. The SPX address is specified in the following form:

```
       network.node(socket)
```

where

network
Is the 4-byte network address of the network on which the remote machine resides,

node
Is the 6-byte node address, which is the LAN address of the LAN adapter in the remote machine

socket
Is the 2-byte socket number on which the remote machine listens.

The default socket number used by WebSphere MQ is 5E86. You can change the default socket number by specifying it in the Windows registry or in the queue manager configuration file qm.ini. The lines in the Windows registry might read:

```
SPX:
  SOCKET=n
```

For more information about values you can set in qm.ini, see “Configuration file stanzas for distributed queuing” on page 53.

The SPX address is later specified in the CONNAME parameter of the sender channel definition. If the WebSphere MQ systems being connected reside on the same network, the network address need not be specified. Similarly, if the remote system is listening on the default socket number (5E86), it need not be specified. A fully qualified SPX address in the CONNAME parameter is:

```
CONNAME('network.node(socket)')
```

but if the systems reside on the same network and the default socket number is used, the parameter is:

```
CONNAME(node)
```

A detailed example of the channel configuration parameters is given in “IBM WebSphere MQ configuration” on page 11.

Receiving on SPX

Receiving channel programs are started in response to a startup request from the sending channel. To do this, a listener program has to be started to detect incoming network requests and start the associated channel.

You should use the WebSphere MQ listener.

Using the WebSphere MQ listener

To run the Listener supplied with WebSphere MQ, that starts new channels as threads, use the RUNMQLSR command. For example:

```
RUNMQLSR -t spx
```

Optionally you can specify the queue manager name or the socket number if you are not using the defaults.

IBM WebSphere MQ 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 WebSphere 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 WebSphere 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.

Default configuration

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

For information about using the Postcard application, see Verify the installation using the Postcard application .

Basic configuration

You can create and start a queue manager from the IBM WebSphere 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.

Channel configuration

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 Figure 1 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 WebSphere MQ for Windows and WebSphere MQ for AIX. To connect to WebSphere 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 WebSphere 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 1. Configuration worksheet for WebSphere MQ for Windows				
	Parameter Name	**Reference**	**Example Used**	**User Value**
Definition for local node				
A	Queue Manager Name		**WINNT**	
B	Local queue name		**WINNT.LOCALQ**	
Connection to WebSphere MQ for AIX The values in this section of the table must match those used in Table 2 on page 19, 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**	
Connection to MQSeries® 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.WINNT.TCP**	
J	Receiver (TCP) channel name	H	**WINNT.DECUX.TCP**	
Connection to WebSphere MQ for HP-UX The values in this section of the table must match those used in Table 3 on page 25, 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**	

Table 1. Configuration worksheet for WebSphere MQ for Windows (continued)				
	Parameter Name	**Reference**	**Example Used**	**User Value**
Connection to WebSphere MQ for Solaris The values in this section of the table must match those used in Table 4 on page 31, 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**	
Connection to WebSphere MQ for Linux The values in this section of the table must match those used in Table 5 on page 37, 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**	
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**	
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**	

Table 1. Configuration worksheet for WebSphere MQ for Windows (continued)				
	Parameter Name	**Reference**	**Example Used**	**User Value**
I	Receiver (SNA) channel name	G	**MVS.WINNT.SNA**	
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**	
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		**WINNT.VSE.SNA**	
I	Receiver channel name	G	**VSE.WINNT.SNA**	

IBM WebSphere 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
```

IBM WebSphere 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
```

IBM WebSphere 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
```

IBM WebSphere 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
```

Automatic startup

WebSphere 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 WebSphere 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 WebSphere MQ service when the system is started.

For more information, see Administering IBM WebSphere MQ.

Running channels as processes or threads

WebSphere 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.

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.

With WebSphere MQ for Windows, use the WebSphere MQ Explorer to set the *PipeLineLength* parameter in the registry.

Note:

1. *PipeLineLength* applies only to V5.2 or later products.

2. 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.

Example configuration - IBM WebSphere MQ for AIX

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

The following platforms are covered in the examples:

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

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

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.

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 WebSphere MQ command used to start the WebSphere 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 WebSphere 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 WebSphere MQ for AIX configuration" on page 18.

IBM WebSphere 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 WebSphere MQ is installed.
5. Error logs are stored in /var/mqm/qmgrs/*qmgrname*/errors.
6. On AIX, you can start a trace of the WebSphere MQ components by using standard WebSphere MQ trace commands, or using AIX system trace. See Using trace for more information about WebSphere 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.

Channel configuration

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 Figure 1 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 WebSphere MQ for AIX and WebSphere MQ for Windows. To connect to WebSphere 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 WebSphere 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 WebSphere MQ for AIX

ID	Parameter Name	Reference	Example Used	User Value
Definition for local node				
A	Queue Manager Name		**AIX**	
B	Local queue name		**AIX.LOCALQ**	
Connection to WebSphere MQ for Windows The values in this section of the table must match those used in Table 1 on page 13, as indicated.				
C	Remote queue manager name	A	**WINNT**	
D	Remote queue name		**WINNT.REMOTEQ**	

Table 2. Configuration worksheet for WebSphere MQ for AIX (continued)				
ID	**Parameter Name**	**Reference**	**Example Used**	**User Value**
E	Queue name at remote system	B	**WINNT.LOCALQ**	
F	Transmission queue name		**WINNT**	
G	Sender (SNA) channel name		**AIX.WINNT.SNA**	
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**	
Connection to WebSphere 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.AIX.TCP**	
J	Receiver (TCP) channel name	H	**AIX.DECUX.TCP**	
Connection to WebSphere MQ for HP-UX The values in this section of the table must match those used in Table 3 on page 25, 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**	
Connection to WebSphere MQ for Solaris The values in this section of the table must match those used in Table 4 on page 31, 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**	

Table 2. Configuration worksheet for WebSphere MQ for AIX (continued)				
ID	**Parameter Name**	**Reference**	**Example Used**	**User Value**
Connection to WebSphere MQ for Linux The values in this section of the table must match those used in Table 5 on page 37, 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**	
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**	
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**	
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**	
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**	
H	Sender (TCP) channel name		**AIX.QSG.TCP**	
I	Receiver (SNA) channel name	G	**QSG.AIX.SNA**	
Connection to MQSeries for VSE/ESA The values in this section of the table must match those used in your VSE/ESA system.				

Table 2. Configuration worksheet for WebSphere MQ for AIX (continued)				
ID	**Parameter Name**	**Reference**	**Example Used**	**User Value**
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		**AIX.VSE.SNA**	
I	Receiver channel name	G	**VSE.AIX.SNA**	

IBM WebSphere MQ 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
```

IBM WebSphere 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
```

IBM WebSphere 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 WebSphere 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.

IBM WebSphere 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
```

IBM WebSphere 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
```

Example configuration - IBM WebSphere MQ for HP-UX

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

The following platforms are included:

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

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

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.

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 WebSphere 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 WebSphere MQ for HP-UX configuration" on page 24.

IBM WebSphere 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 WebSphere 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.

Channel configuration

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 Figure 1 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 WebSphere MQ for HP-UX and WebSphere MQ for Windows. To connect to WebSphere 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 WebSphere 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 3. Configuration worksheet for WebSphere 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 WebSphere MQ for Windows The values in this section of the table must match those used in Table 1 on page 13, 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**	

Table 3. Configuration worksheet for WebSphere MQ for HP-UX (continued)				
ID	**Parameter Name**	**Reference**	**Example Used**	**User Value**
I	Receiver (SNA) channel name	G	**WINNT.HPUX.SNA**	
J	Receiver (TCP) channel name	H	**WINNT.HPUX.TCP**	
Connection to WebSphere MQ for AIX The values in this section of the table must match those used in Table 2 on page 19, 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 WebSphere 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 WebSphere MQ for Solaris The values in this section of the table must match those used in Table 4 on page 31, 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 WebSphere MQ for Linux The values in this section of the table must match those used in Table 5 on page 37, as indicated.				
C	Remote queue manager name	A	**LINUX**	
D	Remote queue name		**LINUX.REMOTEQ**	

Table 3. Configuration worksheet for WebSphere 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**	
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**	
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**	
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**	

IBM WebSphere 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
```

IBM WebSphere 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
```

IBM WebSphere 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 WebSphere MQ is installed.

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

IBM WebSphere 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
```

IBM WebSphere 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
```

Example configuration - IBM WebSphere MQ for Solaris

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

Examples are given on the following platforms:

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

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

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 Oracle documentation: Configuring Intersystem Communications (ISC)

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 WebSphere 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 WebSphere MQ for Solaris configuration" on page 30.

IBM WebSphere 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 WebSphere 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.

Channel configuration

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 WebSphere MQ for Solaris and WebSphere MQ for Windows. To connect to WebSphere 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 WebSphere 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 4. Configuration worksheet for WebSphere 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 WebSphere MQ for Windows The values in this section of the table must match those used in Table 1 on page 13, 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 WebSphere MQ for AIX The values in this section of the table must match those used in Table 2 on page 19, 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**	

Table 4. Configuration worksheet for WebSphere MQ for Solaris (continued)				
ID	**Parameter Name**	**Reference**	**Example Used**	**User Value**
H	Sender (TCP) channel name		**DECUX.SOLARIS.TCP**	
J	Receiver (TCP) channel name	H	**SOLARIS.DECUX.TCP**	
Connection to WebSphere MQ for HP-UX The values in this section of the table must match those used in Table 3 on page 25, 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 WebSphere MQ for Linux The values in this section of the table must match those used in Table 5 on page 37, 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**	
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**	
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**	

Table 4. Configuration worksheet for WebSphere MQ for Solaris (continued)				
ID	**Parameter Name**	**Reference**	**Example Used**	**User Value**
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**	
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**	

IBM WebSphere 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
```

IBM WebSphere 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
```

IBM WebSphere 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
```

IBM WebSphere 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
```

Example configuration - IBM WebSphere MQ for Linux

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

The examples given are on the following platforms:

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

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

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 WebSphere MQ for Linux (x86 platform). It does not apply to WebSphere MQ for Linux (x86-64 platform), WebSphere MQ for Linux (zSeries s390x platform), or WebSphere MQ for Linux (POWER®).

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

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 WebSphere 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 WebSphere 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 WebSphere MQ in the XINETD configuration directory, /etc/xinetd.d. Add the following stanza to the file:

```
# WebSphere 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 WebSphere 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 WebSphere MQ for Linux configuration" on page 36.

IBM WebSphere 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 WebSphere 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.

Channel configuration

The following section details the configuration to be performed on the Linux queue manager 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 WebSphere MQ for Linux and WebSphere MQ for HP-UX. To connect to WebSphere 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 user-specified and reflect the names of WebSphere 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 5. Configuration worksheet for WebSphere MQ for Linux

ID	Parameter Name	Reference	Example Used	User Value
Definition for local node				
A	Queue Manager Name		**LINUX**	
B	Local queue name		**LINUX.LOCALQ**	
Connection to WebSphere MQ for Windows The values in this section of the table must match those used in Table 1 on page 13, 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**	
Connection to WebSphere MQ for AIX The values in this section of the table must match those used in Table 2 on page 19, 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**	
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.				

Table 5. Configuration worksheet for WebSphere MQ for Linux (continued)				
ID	**Parameter Name**	**Reference**	**Example Used**	**User Value**
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.LINUX.TCP**	
J	Receiver (TCP) channel name	H	**LINUX.DECUX.TCP**	
Connection to WebSphere MQ for HP-UX The values in this section of the table must match those used in Table 3 on page 25, 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**	
I	Receiver (SNA) channel name	G	**HPUX.LINUX.SNA**	
J	Receiver (TCP/IP) channel name	H	**HPUX.LINUX.TCP**	
Connection to WebSphere MQ for Solaris The values in this section of the table must match those used in Table 4 on page 31, 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**	
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**	

Table 5. Configuration worksheet for WebSphere MQ for Linux (continued)				
ID	**Parameter Name**	**Reference**	**Example Used**	**User Value**
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**	
Connection to MQSeries for VSE/ESA (WebSphere MQ for Linux (x86 platform) only) 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		**LINUX.VSE.SNA**	
I	Receiver channel name	G	**VSE.LINUX.SNA**	

IBM WebSphere 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
```

IBM WebSphere 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
```

IBM WebSphere 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
```

IBM WebSphere MQ for Linux receiver-channel definitions using TCP/IP
Example coding.

```
def ql (LINUX.LOCALQ) replace                    B
&#xF000;
def chl (HPUX.LINUX.TCP) chltype(rcvr) +         J
    trptype(tcp) +
    replace
```

Queue names

Use this information to understand the restrictions of queue names and reserved queue names.

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 WebSphere 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
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 WebSphere MQ channel authentication data queue
SYSTEM.CICS.INITIATION.QUEUE	Queue used for triggering (not for z/OS)

Queue Name	Description
SYSTEM.CLUSTER.COMMAND.QUEUE	Queue used to communicate repository changes between queue managers (AIX, HP-UX, Linux, IBM i, Solaris, Windows, and z/OS only)
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 (AIX, HP-UX, Linux, IBM i, Solaris, Windows, and z/OS only)
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 (AIX, HP-UX, Linux, IBM i, Solaris, Windows, and z/OS only)
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 WebSphere 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
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 WebSphere 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)

Queue Name	Description
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 WebSphere MQ internal selection evaluation queue (not for z/OS)
SYSTEM.SELECTION.VALIDATION.QUEUE	IBM WebSphere MQ internal selection validation queue (not for z/OS)

Other object names

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 WebSphere 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 (Windows, UNIX and Linux systems only)
SYSTEM.AUTO.SVRCONN	Default server-connection channel for auto definition (IBM i, Windows, UNIX and Linux systems only)
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

Object Name	Description
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 (IBM i, Windows, UNIX and Linux systems only)
SYSTEM.DEFAULT.NAMELIST	Default namelist definition
SYSTEM.DEFAULT.PROCESS	Default process definition
SYSTEM.DEFAULT.SEVICE	Default service (IBM i, Windows, UNIX and Linux systems only)
SYSTEM.DEFAULT.TOPIC	Default topic definition
SYSTEM.QPUBSUB.QUEUE.NAMELIST	A list of queues for the Queued Publish/Subscribe interface to monitor
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

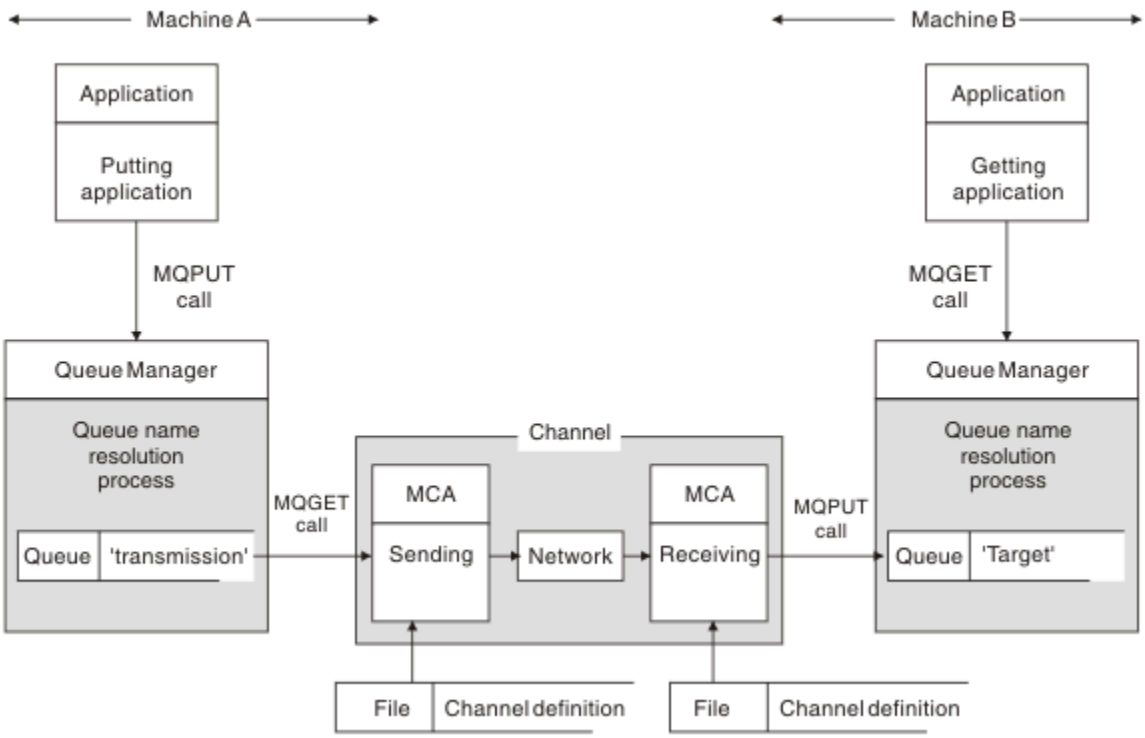

Figure 2. Name resolution

Referring to Figure 2 on page 44, 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 WebSphere 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 6 on page 46 lists the system and default queue objects.
- Table 7 on page 47 lists the system and default topic objects.
- Table 8 on page 47 lists the system and default channel objects.

- Table 9 on page 48 lists the system and default authentication information objects.
- Table 10 on page 48 lists the system and default listener objects.
- Table 11 on page 48 lists the system and default namelist objects.
- Table 12 on page 48 lists the system and default process objects.
- Table 13 on page 48 lists the system and default service objects.

Table 6. 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 WebSphere 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.
SYSTEM.DEFAULT.ALIAS.QUEUE	Default alias queue.

Table 6. System and default objects: queues (continued)	
Object name	**Description**
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 WebSphere MQ Explorer reply-to queue. This is a model queue that creates a temporary dynamic queue for replies to the IBM WebSphere 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 7. 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 8. 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.
SYSTEM.DEF.CLNTCONN	Default client-connection channel.

Table 9. 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 10. 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 11. System and default objects: namelists	
Object name	**Description**
SYSTEM.DEFAULT.NAMELIST	Default namelist.

Table 12. System and default objects: processes	
Object name	**Description**
SYSTEM.DEFAULT.PROCESS	Default process definition.

Table 13. System and default objects: services	
Object name	**Description**
SYSTEM.DEFAULT.SERVICE	Default service.
SYSTEM.BROKER	Publish/subscribe broker

Windows default configuration objects

On Windows systems, you can set up a default configuration using the WebSphere 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 WebSphere MQ object names (for example, hyphens) are replaced by an underscore character.

Valid characters for WebSphere 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 14 on page 49 are created.

Table 14. 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 WebSphere 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+. Once WebSphere 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 WebSphere MQ Postcard application	The local message queue for use by the WebSphere 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 14 on page 49, 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 15. Default values of SYSTEM.BASE.TOPIC

Parameter	Value
TOPICSTR	"
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
SUB	ENABLED
SUBSCOPE	ALL
USEDLQ	YES
WILDCARD	PASSTHRU

If this object does not exist, its default values are still used by IBM WebSphere 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.

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 WebSphere 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:
  - *AccessMode* stanza (Windows only)
  - *Service* stanza - for Installable services
  - *Log* stanza
  - *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
  - *LU62*, *NETBIOS* and *SPX* stanza (Windows only)
  - *MessageBuffer* stanza
  - *SSL* stanza
  - *TCP* stanza

- "Configuration file stanzas for distributed queuing" on page 53 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.

mqs.ini file

Example of an IBM WebSphere 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 WebSphere 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*
- *Service* and *ServiceComponent*

  To configure *InstallableServices*:

  - On UNIX and Linux platforms, use the *Service* and *ServiceComponent* stanzas.

  - On Windows, use **regedit**.
- *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 use the crtmqm command to change these properties:

- *AccessMode* (Windows only)
- *RestrictedMode* (UNIX and Linux systems only)

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 WebSphere MQ on Windows, UNIX and Linux systems. The file is called qm.ini on all platforms.

The stanzas that relate to distributed queuing are:

- CHANNELS
- TCP
- LU62
- NETBIOS
- SPX (Windows XP and Windows 2003 Server only)
- EXITPATH

Figure 3 on page 54 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[1]    ; Whether the binding for applications is to be
                         ; "fastpath" or "standard".
                         ; The default is "standard".
   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".
   TCP:                  ; TCP entries
   PORT=n                ; Port number, the default is 1414
   KEEPALIVE=Yes         ; Switch TCP/IP KeepAlive on
   LIBRARY2=DLLName2     ; Used if code is in two libraries
   EXITPATH:[2]           ; Location of user exits (MQSeries for AIX,
                         ; HP-UX, and Solaris only)
   EXITPATHS=            ; String of directory paths.
```

Figure 3. qm.ini stanzas for distributed queuing

Note:

1. MQIBINDTYPE applies only to IBM WebSphere MQ for AIX, IBM WebSphere MQ for HP-UX, and IBM WebSphere MQ for Solaris.
2. EXITPATH applies only to IBM WebSphere MQ for AIX, IBM WebSphere MQ for HP-UX, and IBM WebSphere MQ for Solaris.

Related information

Configuring

Changing configuration information on Windows, UNIX, and Linux systems

Channel attributes

This section describes the channel attributes held in the channel definitions.

This information is product-sensitive programming interface information.

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.

Note: In WebSphere 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 WebSphere MQ channel attributes are listed in Table 16 on page 55.

Table 16. Channel attributes for the channel types

Attribute field	MQSC command parameter	SDR	SVR	RCVR	RQSTR	CLNT-CONN	SVR-CONN	CLUS - SDR	CLUS-RCVR
Alter date	ALTDATE	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
Alter time	ALTTIME	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
Batch heartbeat interval	BATCHHB	Yes	Yes					Yes	Yes
Batch interval	BATCHINT	Yes	Yes					Yes	Yes
Batch size	BATCHSZ	Yes	Yes	Yes	Yes			Yes	Yes
Channel name	CHANNEL	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
Channel statistics	STATCHL	Yes	Yes	Yes	Yes			Yes	Yes
Channel type	CHLTYPE	Yes	Yes	Yes	Yes	Yes	Yes	Yes	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
Disconnect interval	DISCINT	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
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
LU 6.2 transaction program name	TPNAME	Yes	Yes		Yes	Yes		Yes	Yes
Maximum instances	MAXINST						Yes		
Maximum instances per client	MAXINSTC						Yes		

Table 16. Channel attributes for the channel types (continued)

Attribute field	MQSC command parameter	SDR	SVR	RCVR	RQSTR	CLNT-CONN	SVR-CONN	CLUS - SDR	CLUS-RCVR
Maximum message length	MAXMSGL	Yes	Yes	Yes	Yes	Yes	Yes	Yes	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
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	
Property control	PROPCTL	Yes	Yes					Yes	Yes
PUT authority	PUTAUT			Yes	Yes				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

Table 16. Channel attributes for the channel types (continued)

Attribute field	MQSC command parameter	SDR	SVR	RCVR	RQSTR	CLNT-CONN	SVR-CONN	CLUS - SDR	CLUS-RCVR
SSL Cipher Specification	SSLCIPH	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
SSL Client Authentication	SSLCAUTH		Yes	Yes	Yes		Yes		Yes
SSL Peer	SSLPEER	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
Transmission queue name	XMITQ	Yes	Yes						
Transport type	TRPTYPE	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
Use Dead-Letter Queue	USEDLQ	Yes	Yes	Yes	Yes			Yes	Yes
User ID	USERID	Yes	Yes		Yes	Yes		Yes	

Related concepts

"Channel attributes in alphabetical order" on page 57
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.

WebSphere 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.

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 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 the number of messages specified in BATCHSZ has been sent or when the transmission queue becomes empty. On lightly loaded channels, 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.
- 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 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. However, for data integrity reasons, channels connecting to any of the current platforms must specify a batch size greater than 1. A value of 1 is for use with Version 1 products, apart from WebSphere MQ for MVS.
- 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

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.

Channel statistics (STATCHL)

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.

This attribute is not supported on z/OS.

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 (WebSphere MQ for Windows systems, and UNIX systems only)

  Note: Client-connection channels can also be defined on z/OS for use on other platforms.
- Server-connection

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

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 WebSphere 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

Cluster workload priority (CLWLPRTY)

This attribute specifies the priority of the channel.

The value must be in the range 0 through 9, where 0 is the lowest priority and 9 is the highest.

This attribute is valid for channel types of:

- Cluster sender
- Cluster receiver

Cluster workload rank (CLWLRANK)

This attribute specifies the rank of the channel.

The value must be in the range 0 through 9, where 0 is the lowest rank and 9 is the highest.

This attribute is valid for channel types of:

- Cluster sender
- Cluster receiver

Cluster workload weight (CLWLWGHT)

This attribute applies a weighting factor to the channel so the proportion of messages sent down that channel can be controlled.

The value must be in the range 1 through 99, where 1 is the lowest weighting and 99 is the highest.

This attribute is valid for channel types of:

- Cluster sender
- Cluster receiver

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 WebSphere MQ classes for Java and IBM WebSphere 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 WebSphere MQ classes for Java and IBM WebSphere 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 Version 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 Version 7.0.1.

On AIX, HP-UX, IBM i, Linux, Solaris, and Windows platforms, the TCP/IP connection name parameter of a cluster-receiver channel is optional. If you leave the connection name blank, IBM WebSphere 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 name is up to 48 characters (see note 1) for z/OS, 264 characters for other platforms, and:

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))`. It can include the IP_name of a z/OS 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, Connection name interacts with Local Address to determine which IP stack is used. See “Local Address (LOCLADDR)” on page 68 for further information.

If the transport type is LU 6.2

For WebSphere MQ for IBM i, Windows systems, and UNIX systems, give the fully-qualified name of the partner LU if the TPNAME and MODENAME are specified. For other versions or if the TPNAME and MODENAME are blank, give the CPI-C side information object name for your specific platform.

On 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.

  Note: 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: For cluster-receiver channels, the side information is on the other queue managers in the cluster. Alternatively, 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 WebSphere 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. A workaround to the 48 character limit might be one of the following suggestions:
   - 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 83.

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 67 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.

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.

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 WebSphere 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 WebSphere 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 Version 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. On platforms other than z/OS, 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 67.

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 other platforms, 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 WebSphere MQ Explorer. Keepalive must also be switched on 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 17. Negotiated HBINT value and the corresponding KAINT value

Negotiated HBINT	**KAINT**
>0	Negotiated HBINT + 60 seconds
0	0

If AUTO is specified for KAINT, and it is a server-connection channel, the TCP INTERVAL value is used instead for the keepalive interval.

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.

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 18. 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.

Table 18. Examples of how the LOCLADDR parameter can be used (continued)	
LOCLADDR	**Meaning**
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.

distributed On distributed platforms, it is possible to set a default local address value that will be used for all sender channels that do not have a local address defined. The default value is defined by setting the MQ_LCLADDR environment variable prior to starting the queue manager. The format of the value matches that of MQSC attribute LOCLADDR.

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.

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.

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).

(Retry is not attempted if the cause of failure is such that a retry is not likely to be successful.)

If the channel initiator (on z/OS) or the channel (on distributed platforms) 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 distributed platforms) 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.

Note: For IBM i, UNIX systems, 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 once the first message flows across the channel successfully after the channel went into RUNNING state, that is; once 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.

The *long retry count* attribute can be set from zero through 999 999 999.

This attribute is valid for channel types of:

- Sender
- Server
- Cluster sender
- Cluster receiver

Note: For UNIX systems, 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.

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 Connecting applications using 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 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.

The Client Attachment feature (CAF) is an option of WebSphere MQ for z/OS that supports the attachment of clients to z/OS. If you do not have the Client Attachment feature (CAF) installed, the attribute can be set from zero to five only on the SYSTEM.ADMIN.SVRCONN channel. A value greater than five is interpreted as zero without the CAF installed.

If the value is reduced below 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.

This attribute is valid for server-connection channels only.

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.

The Client Attachment feature (CAF) is an option of WebSphere MQ for z/OS that supports the attachment of clients to z/OS. If you do not have the Client Attachment feature (CAF) installed, the attribute can be set from zero to five only on the SYSTEM.ADMIN.SVRCONN channel. A value greater than five is interpreted as zero without the CAF installed.

If the value is reduced below 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.

Maximum message length (MAXMSGL)

This attribute specifies the maximum length of a message that can be transmitted on the channel.

On WebSphere MQ for UNIX systems, 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.

Because various implementations of WebSphere 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, WebSphere MQ Advanced Message Security increases the length of the message.

Note:

1. You can use a maximum message size of 0 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 WebSphere 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 RUNMSLSR. It runs as a process if you use **`inetd`**.

On WebSphere 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 WebSphere 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 WebSphere 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 79.

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 79.

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 79. 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 switched off.

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 .

Network-connection priority (NETPRTY)

This attribute specifies the priority for the network connection.

Distributed queuing chooses the path with the highest priority if there are multiple paths available. The value must be in the range 0 through 9; 0 is the lowest priority.

This attribute is valid for channel types of:

- Cluster receiver

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.

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 WebSphere 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

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
- 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).

You can choose one of the following:

Process security, also called default authority (DEF)
The default user ID is used.

On all platforms, 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.

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 all platforms, 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.

Context security (CTX) is not supported on server-connection channels.

Further details about context fields and open options can be found in Controlling context information.

More information about security can be found in:

- Security
- Setting up security on Windows, UNIX and Linux systems for WebSphere MQ UNIX systems and Windows systems,

Queue manager name (QMNAME)

This attribute specifies the name of the queue manager or queue manager group to which a WebSphere 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 WebSphere 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 systems, 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 WebSphere MQ for IBM i, you can list up to 10 exit names. In WebSphere 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 WebSphere 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 WebSphere 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 WebSphere 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 79. 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 79.

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 79.

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.

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 terminates.

(Retry is not attempted if the cause of failure is such that a retry is not likely to be successful.)

If the channel initiator (on z/OS) or the channel (on distributed platforms) 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 distributed platforms) 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.

Note: For UNIX systems, 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 once the first message flows across the channel successfully after the channel went into RUNNING state, that is; once 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.

This attribute can be set from zero through 999 999 999.

This attribute is valid for channel types of:

- Sender
- Server
- Cluster sender
- Cluster receiver

Note: On UNIX systems, 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.

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

SSL Cipher Specification (SSLCIPH)

This attribute specifies a single CipherSpec for a TLS or SSL connection.

Every WebSphere 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 SSL or 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 utilize SSL or TLS. Again, in this case, both ends of the channel must specify the same SSLCIPH value.

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 an SSL certificate from an SSL client.

Possible values are:

OPTIONAL
If the peer SSL client sends a certificate, the certificate is processed as normal but authentication does not fail if no certificate is sent.

REQUIRED
If the SSL client does not send a certificate, authentication fails.

The default value is REQUIRED.

You can specify a value for SSLCAUTH on a non-SSL channel definition, one on which SSLCIPH 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 MQSC reference and Security.

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 WebSphere MQ channel.

Note: An alternative way of restricting connections into channels by matching against the SSL or TLS Subject Distinguished Name, is to use channel authentication records. With channel authentication records, different SSL or 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.

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-SSL channel definition, one on which SSLCIPH is missing or blank. You can use this to temporarily disable SSL for debugging without having to clear and later reinput the SSL parameters.

For more information about using SSLPEER, see MQSC reference and Security.

This attribute is valid for all channel types.

Related concepts

Channel authentication records

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 83)
SPX	SPX (“1” on page 83)
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 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 behaviour 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 WebSphere 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 WebSphere 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 WebSphere MQ cluster commands

The IBM WebSphere 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 WebSphere 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.
- 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 WebSphere MQ Explorer

The Explorer GUI can administer a cluster with repository queue managers on IBM WebSphere MQ for z/OS Version 6 or later. You do not need to nominate an additional repository on a separate system. For

earlier versions of WebSphere MQ for z/OS, the IBM WebSphere MQ Explorer cannot administer a cluster with repository queue managers. You must therefore nominate an additional repository on a system that the IBM WebSphere MQ Explorer can administer.

On IBM WebSphere MQ for Windows and WebSphere MQ for Linux, you can also use IBM WebSphere MQ Explorer to work with clusters. You can also use the stand-alone IBM WebSphere MQ Explorer client.

Using the IBM WebSphere MQ Explorer you can view cluster queues and inquire about the status of cluster-sender and cluster-receiver channels. IBM WebSphere 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 19. 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 concepts

“IBM WebSphere MQ cluster commands” on page 84
The IBM WebSphere 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.

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 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, WebSphere 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 WebSphere 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.

On AIX, HP-UX, IBM i, Linux, Solaris, and Windows platforms, the TCP/IP connection name parameter of a cluster-receiver channel is optional. If you leave the connection name blank, IBM WebSphere 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, WebSphere 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, WebSphere MQ regenerates the CONNAME and updates the repositories accordingly.

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 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 Version 7.1, it must not be used if any of the applications opening this queue are connecting to IBM WebSphere MQ Version 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 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 return 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 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:

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.

CLUSDATE
The date at which the definition became available to the local queue manager.

CLUSTIME
The time at which the definition became available to the local queue manager.

STATUS
The status of the cluster-sender channel for this queue manager.

SUSPEND
Whether the queue manager is suspended.

CLUSTER
What clusters the queue manager is in.

CHANNEL
The cluster-receiver channel name for the queue manager.

XMITQ
The cluster transmission queue used by the queue manager. The property is only available on platforms other than z/OS.

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.

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 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: Use of the **REFRESH CLUSTER** command can be disruptive to the cluster while it is in progress, for example by creating a sudden increase in work for the full repositories as they process the repropagation of a queue manager cluster resources. For such reasons it is best to avoid use of the command in day-to-day work if possible and use alternative methods to correct specific inconsistencies.

Related concepts

Clustering: REFRESH CLUSTER and the history queue

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.

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 WebSphere 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 issue the REFRESH QMGR TYPE(PROXYSUB) command to reinstate any required proxy subscriptions. See REFRESH CLUSTER considerations for publish/subscribe clusters.

Using the RESET CLUSTER command is the only way to delete auto-defined cluster-sender channels. 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 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

If a cluster contains more than one instance of the same queue, WebSphere MQ selects a queue manager to route a message to. It uses the cluster workload management algorithm to determine the best queue manager to use. You can provide the workload balancing algorithm to select the queue manager by writing a cluster workload exit program.

Suitable destinations are chosen based on the availability of the queue manager and queue, and on a number of cluster workload-specific attributes associated with queue managers, queues and channels.

If the results of the workload balancing algorithm do not meet your needs, you can write a cluster workload user exit program. Use the exit to route messages to the queue of your choice in the cluster.

Related concepts

Cluster workload exit call and data structures
This section provides reference information for the cluster workload exit and the data structures. This is general-use programming interface information.

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.

This section lists 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 20. Attributes for cluster workload management

Queues	Queue managers	Channels
• CLWLPRTY • CLWLRANK • CLWLUSEQ	• CLWLUSEQ • CLWLMRUC	• CLWLPRTY • CLWLRANK • CLWLWGHT • NETPRTY

Initially, the queue manager builds a list of possible destinations from two procedures:

- 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, or in other words, 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 steps of the algorithm that follow eliminate destinations from the list of possible destinations.

1. If a queue name is specified:
   a. Queues that are not put enabled are eliminated as possible destinations.
   b. Remote instances of queues that do not share a cluster with the local queue manager are eliminated.
   c. Remote CLUSRCVR channels that are not in the same cluster as the queue are eliminated.
2. If a queue manager name is specified,
   a. Queue manager aliases that are not put enabled are eliminated.
   b. Remote CLUSRCVR channels that are not in the same cluster as the local queue manager are eliminated.
3. If the resulting set of queues contains the local instance of the queue, the local instance of a queue is typically used. The local instance of the queue is used if one of these three conditions are true:
   - The use-queue attribute of the queue, CLWLUSEQ is set to LOCAL.
   - Both the following are true:
     a. The use-queue attribute of the queue, CLWLUSEQ is set to QMGR.
     b. 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.

     Note: You can detect a message from a cluster channel in a user exit if the both the MQWXP_PUT_BY_CLUSTER_CH and MQQF_CLWL_USEQ_ANY flags are not set:

     – MQWXP.Flags flag MQWXP_PUT_BY_CLUSTER_CH.
     – MQWQR.QFlags flag MQQF_CLWL_USEQ_ANY.
4. If the message is a cluster PCF message, any queue manager to which a publication or subscription has already been sent is eliminated.
5. All channels to queue managers or queue manager alias with a CLWLRANK less than the maximum rank of all remaining channels or queue manager aliases are eliminated.
6. All queues (not queue manager aliases) with a CLWLRANK less than the maximum rank of all remaining queues are eliminated.
7. If only remote instances of a queue remain, resumed queue managers are chosen in preference to suspended ones.
8. If more than one remote instance of a queue 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 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 remains, all channels that are being tried again, MQCHS_RETRYING are included.
11. If no remote instance of a queue remains, all channels in requesting, paused, or stopped state are included. The state constants are listed:
    - MQCHS_REQUESTING
    - MQCHS_PAUSED
    - MQCHS_STOPPED
12. If more than one remote instance of a queue remains and the message is a cluster PCF message, locally defined CLUSSDR channels are chosen.
13. If more than one remote instance of a queue to any queue manager remains, channels with the highest NETPRTY value for each queue manager are chosen.
14. If a queue manager is being chosen:
    - 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.
15. If a queue is being chosen:
    - All queues other than queues with the highest priority, CLWLPRTY, are eliminated, and channels are kept.
16. All channels, except a number of channels with the highest values in MQWDR.DestSeqNumber are eliminated. The elimination stops when the number of remaining channels is no greater than the maximum allowed number of most recently used channels, CLWLMRUC.
17. If more than one remote instance of a queue remains, the least recently used channel is chosen. The least recently used channel has the lowest value of MQWDR.DestSeqFactor.
    - If there is more than one channel with the lowest value, one of the channels with the lowest value in MQWDR.DestSeqNumber is chosen.
    - The destination sequence factor of the choice is increased by the queue manager, by approximately 1000/CLWLWGHT.

    Note:

    a. The destination sequence factors of all destinations are reset to zero if the cluster workload attributes of available CLUSRCVR channels are altered. The sequence factors are zeroed if new CLUSRCVR channels become available.

    b. Modifications to workload attributes of manually defined CLUSSDR channels do not reset the Destination Sequence Factor.

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 which 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

CLWLPRTY queue attribute
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.

CLWLRANK queue attribute
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.

CLWLUSEQ queue attribute
The CLWLUSEQ queue attribute specifies whether a local instance of a queue is given preference as a destination over other instances in a cluster.

CLWLUSEQ queue manager attribute

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.

CLWLMRUC queue manager attribute
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.

CLWLPRTY channel attribute
The CLWLPRTY channel attribute specifies the priority of CLUSSDR or CLUSRCVR 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.

CLWLRANK channel attribute
The CLWLRANK channel attribute specifies the rank of CLUSSDR or CLUSRCVR 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.

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.

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.

CLWLPRTY queue attribute

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. WebSphere 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. 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.

WebSphere 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 backup.

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

CLWLRANK queue attribute
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.

CLWLUSEQ queue attribute

The CLWLUSEQ queue attribute specifies whether a local instance of a queue is given preference as a destination over other instances in a cluster.

CLWLUSEQ queue manager attribute
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.

CLWLMRUC queue manager attribute
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.

CLWLPRTY channel attribute
The CLWLPRTY channel attribute specifies the priority of CLUSSDR or CLUSRCVR 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.

CLWLRANK channel attribute
The CLWLRANK channel attribute specifies the rank of CLUSSDR or CLUSRCVR 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.

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.

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.

CLWLRANK queue attribute

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.

WebSphere 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 WebSphere 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.

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

CLWLPRTY queue attribute
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.

CLWLUSEQ queue attribute
The CLWLUSEQ queue attribute specifies whether a local instance of a queue is given preference as a destination over other instances in a cluster.

CLWLUSEQ queue manager attribute
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.

CLWLMRUC queue manager attribute
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.

CLWLPRTY channel attribute
The CLWLPRTY channel attribute specifies the priority of CLUSSDR or CLUSRCVR 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.

CLWLRANK channel attribute
The CLWLRANK channel attribute specifies the rank of CLUSSDR or CLUSRCVR 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.

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.

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.

CLWLUSEQ queue attribute

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

CLWLPRTY queue attribute
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.

CLWLRANK queue attribute
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.

CLWLUSEQ queue manager attribute
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.

CLWLMRUC queue manager attribute
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.

CLWLPRTY channel attribute
The CLWLPRTY channel attribute specifies the priority of CLUSSDR or CLUSRCVR 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.

CLWLRANK channel attribute
The CLWLRANK channel attribute specifies the rank of CLUSSDR or CLUSRCVR 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.

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.

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.

CLWLUSEQ queue manager attribute

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

CLWLPRTY queue attribute

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.

CLWLRANK queue attribute
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.

CLWLUSEQ queue attribute
The CLWLUSEQ queue attribute specifies whether a local instance of a queue is given preference as a destination over other instances in a cluster.

CLWLMRUC queue manager attribute
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.

CLWLPRTY channel attribute
The CLWLPRTY channel attribute specifies the priority of CLUSSDR or CLUSRCVR 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.

CLWLRANK channel attribute
The CLWLRANK channel attribute specifies the rank of CLUSSDR or CLUSRCVR 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.

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.

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.

CLWLMRUC queue manager attribute

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.

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

CLWLPRTY queue attribute
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.

CLWLRANK queue attribute
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.

CLWLUSEQ queue attribute
The CLWLUSEQ queue attribute specifies whether a local instance of a queue is given preference as a destination over other instances in a cluster.

CLWLUSEQ queue manager attribute

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.

CLWLPRTY channel attribute
The CLWLPRTY channel attribute specifies the priority of CLUSSDR or CLUSRCVR 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.

CLWLRANK channel attribute
The CLWLRANK channel attribute specifies the rank of CLUSSDR or CLUSRCVR 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.

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.

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.

CLWLPRTY channel attribute

The CLWLPRTY channel attribute specifies the priority of CLUSSDR or CLUSRCVR 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 preference for a CLUSSDR or CLUSRCVR channel. IBM WebSphere 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.

WebSphere MQ obtains the priority of channels 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 backup.

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

CLWLPRTY queue attribute
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.

CLWLRANK queue attribute
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.

CLWLUSEQ queue attribute

The CLWLUSEQ queue attribute specifies whether a local instance of a queue is given preference as a destination over other instances in a cluster.

CLWLUSEQ queue manager attribute
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.

CLWLMRUC queue manager attribute
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.

CLWLRANK channel attribute
The CLWLRANK channel attribute specifies the rank of CLUSSDR or CLUSRCVR 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.

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.

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.

CLWLRANK channel attribute

The CLWLRANK channel attribute specifies the rank of CLUSSDR or CLUSRCVR 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.

WebSphere MQ obtains the rank of channels before checking channel status. Obtaining 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.

If you used the priority attribute WebSphere 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.

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

CLWLPRTY queue attribute
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.

CLWLRANK queue attribute
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.

CLWLUSEQ queue attribute

The CLWLUSEQ queue attribute specifies whether a local instance of a queue is given preference as a destination over other instances in a cluster.

CLWLUSEQ queue manager attribute
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.

CLWLMRUC queue manager attribute
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.

CLWLPRTY channel attribute
The CLWLPRTY channel attribute specifies the priority of CLUSSDR or CLUSRCVR 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.

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.

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.

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.

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

CLWLPRTY queue attribute
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.

CLWLRANK queue attribute
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.

CLWLUSEQ queue attribute
The CLWLUSEQ queue attribute specifies whether a local instance of a queue is given preference as a destination over other instances in a cluster.

CLWLUSEQ queue manager attribute
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.

CLWLMRUC queue manager attribute
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.

CLWLPRTY channel attribute

The CLWLPRTY channel attribute specifies the priority of CLUSSDR or CLUSRCVR 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.

CLWLRANK channel attribute
The CLWLRANK channel attribute specifies the rank of CLUSSDR or CLUSRCVR 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.

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.

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.

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

CLWLPRTY queue attribute
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.

CLWLRANK queue attribute
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.

CLWLUSEQ queue attribute
The CLWLUSEQ queue attribute specifies whether a local instance of a queue is given preference as a destination over other instances in a cluster.

CLWLUSEQ queue manager attribute
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.

CLWLMRUC queue manager attribute
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.

CLWLPRTY channel attribute
The CLWLPRTY channel attribute specifies the priority of CLUSSDR or CLUSRCVR 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.

CLWLRANK channel attribute
The CLWLRANK channel attribute specifies the rank of CLUSSDR or CLUSRCVR 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.

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.

Cluster workload exit call and data structures

This section provides reference information for the cluster workload exit and the data structures. This is general-use programming interface information.

You can write cluster workload exits in the following programming languages:

- C
- System/390® assembler (WebSphere MQ for z/OS)

The call is described in:

- "MQ_CLUSTER_WORKLOAD_EXIT - Call description" on page 108

The structure data types used by the exit are described in:

- "MQXCLWLN - Navigate Cluster workload records" on page 109
- "MQWXP - Cluster workload exit parameter structure" on page 113
- "MQWDR - Cluster workload destination record structure" on page 120
- "MQWQR - Cluster workload queue record structure" on page 124
- "MQWCR - Cluster workload cluster record structure" on page 129
-

Throughout this section, queue-manager attributes and queue attributes are shown in full. The equivalent names that are used in the MQSC commands book are shown below. For details of MQSC commands, see MQSC reference .

Table 21. Queue-manager attributes

Full name	**Name used in MQSC**
ClusterWorkloadData	CLWLDATA
ClusterWorkloadExit	CLWLEXIT
ClusterWorkloadLength	CLWLLEN

Table 22. Queue attributes

Full name	**Name used in MQSC**
DefBind	DEFBIND
DefPersistence	DEFPSIST
DefPriority	DEFPRTY
InhibitPut	PUT
QDesc	DESCR

Related concepts

Workload balancing

If a cluster contains more than one instance of the same queue, WebSphere MQ selects a queue manager to route a message to. It uses the cluster workload management algorithm to determine the best queue

manager to use. You can provide the workload balancing algorithm to select the queue manager by writing a cluster workload exit program.

MQ_CLUSTER_WORKLOAD_EXIT - Call description

The cluster workload exit is called by the queue manager to route a message to an available queue manager.

Note: No entry point called MQ_CLUSTER_WORKLOAD_EXIT is provided by the queue manager. Instead, the name of the cluster workload exit is defined by the ClusterWorkloadExit queue-manager attribute.

The MQ_CLUSTER_WORKLOAD_EXIT exit is supported on all platforms.

Syntax

```
MQ_CLUSTER_WORKLOAD_EXIT (ExitParms)
```

Related reference

MQXCLWLN - Navigate Cluster workload records
The MQXCLWLN call is used to navigate through the chains of MQWDR, MQWQR, and MQWCR records stored in the cluster cache.

MQWXP - Cluster workload exit parameter structure
The following table summarizes the fields in the MQWXP - Cluster workload exit parameter structure.

MQWDR - Cluster workload destination record structure
The following table summarizes the fields in the MQWDR - Cluster workload destination record structure.

MQWQR - Cluster workload queue record structure
The following table summarizes the fields in the MQWQR - Cluster workload queue record structure.

MQWCR - Cluster workload cluster record structure
The following table summarizes the fields in the MQWCR cluster workload record structure.

Parameters for MQ_CLUSTER_WORKLOAD_EXIT

Description of the parameters in the MQ_CLUSTER_WORKLOAD_EXIT call.

***ExitParms*(MQWXP) - input/output**
Exit parameter block.

- The exit sets information in MQWXP to indicate how to manage the workload.

Related reference

Usage notes
The function performed by the cluster workload exit is defined by the provider of the exit. The exit, however, must conform to the rules defined in the associated control block MQWXP.

Language invocations for MQ_CLUSTER_WORKLOAD_EXIT
The MQ_CLUSTER_WORKLOAD_EXIT supports two languages, C and High Level Assembler.

Usage notes

The function performed by the cluster workload exit is defined by the provider of the exit. The exit, however, must conform to the rules defined in the associated control block MQWXP.

No entry point called MQ_CLUSTER_WORKLOAD_EXIT is provided by the queue manager. However, a typedef is provided for the name MQ_CLUSTER_WORKLOAD_EXIT in the C programming language. Use the typedef to declare the user-written exit, to ensure that the parameters are correct.

Related reference

Parameters for MQ_CLUSTER_WORKLOAD_EXIT

Description of the parameters in the MQ_CLUSTER_WORKLOAD_EXIT call.

Language invocations for MQ_CLUSTER_WORKLOAD_EXIT
The MQ_CLUSTER_WORKLOAD_EXIT supports two languages, C and High Level Assembler.

Language invocations for MQ_CLUSTER_WORKLOAD_EXIT

The MQ_CLUSTER_WORKLOAD_EXIT supports two languages, C and High Level Assembler.

C invocation

```
MQ_CLUSTER_WORKLOAD_EXIT (&ExitParms);
```

Replace *MQ_CLUSTER_WORKLOAD_EXIT* with the name of your cluster workload exit function.

Declare the *MQ_CLUSTER_WORKLOAD_EXIT* parameters as follows:

```
MQWXP  ExitParms;  /* Exit parameter block */
```

High Level Assembler invocation

```
         CALL EXITNAME,(EXITPARMS)
```

Declare the parameters as follows:

```
EXITPARMS           CMQWXPA             Exit parameter block
```

Related reference

Parameters for MQ_CLUSTER_WORKLOAD_EXIT
Description of the parameters in the MQ_CLUSTER_WORKLOAD_EXIT call.

Usage notes
The function performed by the cluster workload exit is defined by the provider of the exit. The exit, however, must conform to the rules defined in the associated control block MQWXP.

MQXCLWLN - Navigate Cluster workload records

The MQXCLWLN call is used to navigate through the chains of MQWDR, MQWQR, and MQWCR records stored in the cluster cache.

The cluster cache is an area of main storage used to store information relating to the cluster.

If the cluster cache is static, it has a fixed size. If you set it to dynamic, the cluster cache can expand as required.

Set the type of cluster cache to STATIC or DYNAMIC using either a system parameter or macro.

- The system parameter ClusterCacheType on platforms other than z/OS
- The CLCACHE parameter in the CSQ6SYSP macro on z/OS.

Syntax

```
MQXCLWLN (ExitParms, CurrentRecord, NextOffset, NextRecord, Compcode, Reason)
```

Related reference

MQ_CLUSTER_WORKLOAD_EXIT - Call description
The cluster workload exit is called by the queue manager to route a message to an available queue manager.

MQWXP - Cluster workload exit parameter structure
The following table summarizes the fields in the MQWXP - Cluster workload exit parameter structure.

MQWDR - Cluster workload destination record structure

The following table summarizes the fields in the MQWDR - Cluster workload destination record structure.

MQWQR - Cluster workload queue record structure
The following table summarizes the fields in the MQWQR - Cluster workload queue record structure.

MQWCR - Cluster workload cluster record structure
The following table summarizes the fields in the MQWCR cluster workload record structure.

Parameters for MQXCLWLN - Navigate Cluster workload records

Description of the parameters in the MQXCLWLN call.

***ExitParms* (MQWXP) - input/output**
Exit parameter block.

This structure contains information relating to the invocation of the exit. The exit sets information in this structure to indicate how to manage the workload.

***CurrentRecord* (MQPTR) - input**
Address of current record.

This structure contains information relating to the address of the record currently being examined by the exit. The record must be one of the following types:

- Cluster workload destination record (MQWDR)
- Cluster workload queue record (MQWQR)
- Cluster workload cluster record (MQWCR)

***NextOffset* (MQLONG) - input**
Offset of next record.

This structure contains information relating to the offset of the next record or structure. *NextOffset* is the value of the appropriate offset field in the current record, and must be one of the following fields:

- ChannelDefOffset field in MQWDR
- ClusterRecOffset field in MQWDR
- ClusterRecOffset field in MQWQR
- ClusterRecOffset field in MQWCR

***NextRecord* (MQPTR) - output**
Address of next record or structure.

This structure contains information relating to the address of the next record or structure. If *CurrentRecord* is the address of an MQWDR, and *NextOffset* is the value of the ChannelDefOffset field, *NextRecord* is the address of the channel definition structure (MQCD).

If there is no next record or structure, the queue manager sets *NextRecord* to the null pointer, and the call returns completion code MQCC_WARNING and reason code MQRC_NO_RECORD_AVAILABLE.

***CompCode* (MQLONG) - output**
Completion code.

The completion code has one of the following values:

MQCC_OK
Successful completion.

MQCC_WARNING
Warning (partial completion).

MQCC_FAILED
Call failed.

***Reason* (MQLONG) - output**
Reason code qualifying CompCode

If CompCode is MQCC_OK:

MQRC_NONE
(0, X'0000')
No reason to report.

If *CompCode* is MQCC_WARNING:

MQRC_NO_RECORD_AVAILABLE
(2359, X'0937')
No record available. An MQXCLWLN call was issued from a cluster workload exit to obtain the address of the next record in the chain. The current record is the last record in the chain. Corrective action: None.

If *CompCode* is MQCC_FAILED:

MQRC_CURRENT_RECORD_ERROR
(2357, X'0935')
CurrentRecord parameter not valid. An MQXCLWLN call was issued from a cluster workload exit to obtain the address of the next record in the chain. The address specified by the *CurrentRecord* parameter is not the address of a valid record.

CurrentRecord must be the address of a destination record, MQWDR, queue record (MQWQR), or cluster record (MQWCR) residing within the cluster cache. Corrective action: Ensure that the cluster workload exit passes the address of a valid record residing in the cluster cache.

MQRC_ENVIRONMENT_ERROR
(2012, X'07DC')
Call not valid in environment. An MQXCLWLN call was issued, but not from a cluster workload exit.

MQRC_NEXT_OFFSET_ERROR
(2358, X'0936')
NextOffset parameter not valid. An MQXCLWLN call was issued from a cluster workload exit to obtain the address of the next record in the chain. The offset specified by the *NextOffset* parameter is not valid. *NextOffset* must be the value of one of the following fields:

- ChannelDefOffset field in MQWDR
- ClusterRecOffset field in MQWDR
- ClusterRecOffset field in MQWQR
- ClusterRecOffset field in MQWCR

Corrective action: Ensure that the value specified for the *NextOffset* parameter is the value of one of the fields listed previously.

MQRC_NEXT_RECORD_ERROR
(2361, X'0939')
NextRecord parameter not valid.

MQRC_WXP_ERROR
(2356, X'0934')
Workload exit parameter structure not valid. An MQXCLWLN call was issued from a cluster workload exit to obtain the address of the next record in the chain. The workload exit parameter structure *ExitParms* is not valid, for one of the following reasons:

- The parameter pointer is not valid. It is not always possible to detect parameter pointers that are not valid; if not detected, unpredictable results occur.
- The StrucId field is not MQWXP_STRUC_ID.
- The Version field is not MQWXP_VERSION_2.
- The Context field does not contain the value passed to the exit by the queue manager.

Corrective action: Ensure that the parameter specified for *ExitParms* is the MQWXP structure that was passed to the exit when the exit was invoked.

Related reference

Usage notes for MQXCLWLN - Navigate Cluster workload records
Use MQXCLWLN to navigate through cluster records, even if the cache is static.

Language invocations of MQXCLWLN
MQXCLWLN supports two languages, C and High Level Assembler.

Usage notes for MQXCLWLN - Navigate Cluster workload records

Use MQXCLWLN to navigate through cluster records, even if the cache is static.

If the cluster cache is dynamic, the MQXCLWLN call must be used to navigate through the records. The exit ends abnormally if simple pointer-and-offset arithmetic is used to navigate through the records.

If the cluster cache is static, MQXCLWLN need not be used to navigate through the records. Typically use MQXCLWLN even when the cache is static. You can then change the cluster cache to being dynamic without needing to change the workload exit.

Related reference

Parameters for MQXCLWLN - Navigate Cluster workload records
Description of the parameters in the MQXCLWLN call.

Language invocations of MQXCLWLN
MQXCLWLN supports two languages, C and High Level Assembler.

Language invocations of MQXCLWLN

MQXCLWLN supports two languages, C and High Level Assembler.

C invocation

```
MQXCLWLN (&ExitParms, CurrentRecord, NextOffset, &NextRecord, &CompCode, &Reason) ;
```

Declare the parameters as follows:

```
Typedef struct tagMQXCLWLN {
  MQWXP     ExitParms;            /* Exit parameter block */
  MQPTR     CurrentRecord;        /* Address of current record*/
  MQLONG    NextOffset;           /* Offset of next record */
  MQPTR     NextRecord;           /* Address of next record or structure */
  MQLONG    CompCode;             /* Completion code */
  MQLONG    Reason;               /* Reason code qualifying CompCode */
```

High Level Assembler invocation

```
CALL MQXCLWLN,(CLWLEXITPARMS,CURRENTRECORD,NEXTOFFSET,NEXTRECORD,COMPCODE,REASON)
```

Declare the parameters as follows:

```
CLWLEXITPARMS  CMQWXPA,      Cluster workload exit parameter block
CURRENTRECORD  CMQWDRA,      Current record
NEXTOFFSET     DS  F         Next offset
NEXTRECORD     DS  F         Next record
COMPCODE       DS  F         Completion code
REASON         DS  F         Reason code qualifying COMPCODE
```

Related reference

Parameters for MQXCLWLN - Navigate Cluster workload records
Description of the parameters in the MQXCLWLN call.

Usage notes for MQXCLWLN - Navigate Cluster workload records

Use MQXCLWLN to navigate through cluster records, even if the cache is static.

MQWXP - Cluster workload exit parameter structure

The following table summarizes the fields in the MQWXP - Cluster workload exit parameter structure.

Table 23. Fields in MQWXP

Field	Description	Page
StrucId	Structure identifier	StrucId
Version	Structure version number	Version
ExitId	Type of exit	ExitId
ExitReason	Reason for invoking exit	ExitReason
ExitResponse	Response from exit	ExitResponse
ExitResponse2	Secondary response from exit	ExitResponse2
Feedback	Feedback code	Feedback
Flags	Flags values. These bit flags are used to indicate information about the message being put	Flags
ExitUserArea	Exit user area	ExitUserArea
ExitData	Exit data	ExitData
MsgDescPtr	Address of message descriptor (MQMD)	MsgDescPtr
MsgBufferPtr	Address of buffer containing some or all the message data	MsgBufferPtr
MsgBufferLength	Length of buffer containing message data	MsgBufferLength
MsgLength	Length of complete message	MsgLength
QName	Name of queue	QName
QMgrName	Name of local queue manager	QMgrName
DestinationCount	Number of possible destinations	DestinationCount
DestinationChosen	Destination chosen	DestinationChosen
DestinationArrayPtr	Address of an array of pointers to destination records (MQWDR)	DestinationArrayPtr
QArrayPtr	Address of an array of pointers to queue records (MQWQR)	QArrayPtr
Note: The remaining fields are ignored if Version is less than MQWXP_VERSION_2.		
CacheContext	Context information	CacheContext
CacheType	Type of cluster cache	CacheType
Note: The remaining fields are ignored if Version is less than MQWXP_VERSION_3.		
CLWLMRUChannels	Maximum number of allowed active outbound cluster channels	CLWLMRUChannels
Note: The remaining fields are ignored if Version is less than MQWXP_VERSION_4.		

Table 23. Fields in MQWXP (continued)		
Field	**Description**	**Page**
pEntryPoints	Address of the MQIEP structure to allow MQI and DCI calls to be made	pEntryPoints

The cluster workload exit parameter structure describes the information that is passed to the cluster workload exit.

The cluster workload exit parameter structure is supported on all platforms

Additionally, the MQWXP1, MQWXP2 and MQWXP3 structures are available for backwards compatibility.

Related reference

MQ_CLUSTER_WORKLOAD_EXIT - Call description
The cluster workload exit is called by the queue manager to route a message to an available queue manager.

MQXCLWLN - Navigate Cluster workload records
The MQXCLWLN call is used to navigate through the chains of MQWDR, MQWQR, and MQWCR records stored in the cluster cache.

MQWDR - Cluster workload destination record structure
The following table summarizes the fields in the MQWDR - Cluster workload destination record structure.

MQWQR - Cluster workload queue record structure
The following table summarizes the fields in the MQWQR - Cluster workload queue record structure.

MQWCR - Cluster workload cluster record structure
The following table summarizes the fields in the MQWCR cluster workload record structure.

Fields in MQWXP - Cluster workload exit parameter structure

Description of the fields in the MQWXP - Cluster workload exit parameter structure

StrucId (MQCHAR4) - input
The structure identifier for the cluster workload exit parameter structure.

- The StrucId value is MQWXP_STRUC_ID.
- For the C programming language, the constant MQWXP_STRUC_ID_ARRAY is also defined. It has the same value as MQWXP_STRUC_ID. It is an array of characters instead of a string.

Version (MQLONG) - input
Indicates the structure version number. Version takes one of the following values:

MQWXP_VERSION_1
Version-1 cluster workload exit parameter structure.

MQWXP_VERSION_1 is supported in all environments.

MQWXP_VERSION_2
Version-2 cluster workload exit parameter structure.

MQWXP_VERSION_2 is supported in the following environments: AIX, HP-UX, Linux, IBM i, Solaris and Windows.

MQWXP_VERSION_3
Version-3 cluster workload exit parameter structure.

MQWXP_VERSION_3 is supported in the following environments: AIX, HP-UX, Linux, IBM i, Solaris and Windows.

MQWXP_VERSION_4
Version-4 cluster workload exit parameter structure.

MQWXP_VERSION_4 is supported in the following environments: AIX, HP-UX, Linux, IBM i, Solaris and Windows.

MQWXP_CURRENT_VERSION
Current version of cluster workload exit parameter structure.

ExitId (MQLONG) - input
Indicates the type of exit being called. The cluster workload exit is the only supported exit.

- The ExitId value must be MQXT_CLUSTER_WORKLOAD_EXIT

ExitReason (MQLONG) - input
Indicates the reason for invoking the cluster workload exit. ExitReason takes one of the following values:

MQXR_INIT
Indicates that the exit is being invoked for the first time.

Acquire and initialize any resources that the exit might need, such as main storage.

MQXR_TERM
Indicates that the exit is about to be terminated.

Free any resources that the exit might have acquired since it was initialized, such as main storage.

MQXR_CLWL_OPEN
Called by MQOPEN.

MQXR_CLWL_PUT
Called by MQPUT or MQPUT1.

MQXR_CLWL_MOVE
Called by MCA when the channel state has changed.

MQXR_CLWL_REPOS
Called by MQPUT or MQPUT1 for a repository-manager PCF message.

MQXR_CLWL_REPOS_MOVE
Called by MCA for a repository-manager PCF message if the channel state has changed.

ExitResponse (MQLONG) - output
Set ExitResponse to indicate whether processing of the message continues. It must be one of the following values:

MQXCC_OK
Continue processing the message normally.

- DestinationChosen identifies the destination to which the message is to be sent.

MQXCC_SUPPRESS_FUNCTION
Discontinue processing the message.

- The actions taken by the queue manager depend on the reason the exit was invoked:

Table 24. Actions taken by the queue manager

ExitReason	Action taken
– MQXR_CLWL_OPEN – MQXR_CLWL_REPOS – MQXR_CLWL_PUT	MQOPEN, MQPUT, or MQPUT1 call fail with completion code MQCC_FAILED and reason code MQRC_STOPPED_BY_CLUSTER_EXIT.
– MQXR_CLWL_MOVE – MQXR_CLWL_REPOS_MOVE	The message is placed on the dead-letter queue.

MQXCC_SUPPRESS_EXIT
Continue processing the current message normally. Do not invoke the exit again until the queue manager shuts down.

The queue manager processes subsequent messages as if the ClusterWorkloadExit queue-manager attribute is blank. DestinationChosen identifies the destination to which the current message is sent.

Any other value
Process the message as if MQXCC_SUPPRESS_FUNCTION is specified.

ExitResponse2 (MQLONG) - input/output
Set ExitResponse2 to provide the queue manager with more information.

- MQXR2_STATIC_CACHE is the default value, and is set on entry to the exit.
- When ExitReason has the value MQXR_INIT, the exit can set one of the following values in ExitResponse2:

  MQXR2_STATIC_CACHE
  The exit requires a static cluster cache.

  – If the cluster cache is static, the exit need not use the MQXCLWLN call to navigate the chains of records in the cluster cache.

  – If the cluster cache is dynamic, the exit cannot navigate correctly through the records in the cache.

  Note: The queue manager processes the return from the MQXR_INIT call as though the exit had returned MQXCC_SUPPRESS_EXIT in the ExitResponse field.

  MQXR2_DYNAMIC_CACHE
  The exit can operate with either a static or dynamic cache.

  – If the exit returns this value, the exit must use the MQXCLWLN call to navigate the chains of records in the cluster cache.

Feedback (MQLONG) - input
A reserved field. The value is zero.

Flags (MQLONG) - input
Indicates information about the message being put.

- The value of Flags is MQWXP_PUT_BY_CLUSTER_CHL. The message originates from a cluster channel, rather than locally or from a non-cluster channel. In other words, the message has come from another cluster queue manager.

Reserved (MQLONG) - input
A reserved field. The value is zero.

ExitUserArea (MQBYTE16) - input/output
Set ExitUserArea to communicate between calls to the exit.

- ExitUserArea is initialized to binary zero before the first invocation of the exit. Any changes made to this field by the exit are preserved across the invocations of the exit that occur between the MQCONN call and the matching MQDISC call. The field is reset to binary zero when the MQDISC call occurs.
- The first invocation of the exit is indicated by the ExitReason field having the value MQXR_INIT.
- The following constants are defined:

  MQXUA_NONE - string
  MQXUA_NONE_ARRAY - character array
  No user information. Both constants are binary zero for the length of the field.

  MQ_EXIT_USER_AREA_LENGTH
  The length of ExitUserArea.

ExitData (MQCHAR32) - input
The value of the ClusterWorkloadData queue-manager attribute. If no value has been defined for that attribute, this field is all blanks.

- The length of ExitData is given by MQ_EXIT_DATA_LENGTH.

MsgDescPtr (PMQMD) - input
The address of a copy of the message descriptor (MQMD) for the message being processed.

- Any changes made to the message descriptor by the exit are ignored by the queue manager.
- If ExitReason has one of the following values MsgDescPtr is set to the null pointer, and no message descriptor is passed to the exit:
  - MQXR_INIT
  - MQXR_TERM
  - MQXR_CLWL_OPEN

MsgBufferPtr (PMQVOID) - input
The address of a buffer containing a copy of the first MsgBufferLength bytes of the message data.

- Any changes made to the message data by the exit are ignored by the queue manager.
- No message data is passed to the exit when:
  - MsgDescPtr is the null pointer.
  - The message has no data.
  - The ClusterWorkloadLength queue-manager attribute is zero.

  In these cases, MsgBufferPtr is the null pointer.

MsgBufferLength (MQLONG) - input
The length of the buffer containing the message data passed to the exit.

- The length is controlled by the ClusterWorkloadLength queue-manager attribute.
- The length might be less than the length of the complete message, see MsgLength.

MsgLength (MQLONG) - input
The length of the complete message passed to the exit.

- MsgBufferLength might be less than the length of the complete message.
- MsgLength is zero if ExitReason is MQXR_INIT, MQXR_TERM, or MQXR_CLWL_OPEN.

QName (MQCHAR48) - input
The name of the destination queue. The queue is a cluster queue.

- The length of QName is MQ_Q_NAME_LENGTH.

QMgrName (MQCHAR48) - input
The name of the local queue manager that has invoked the cluster workload exit.

- The length of QMgrName is MQ_Q_MGR_NAME_LENGTH.

DestinationCount (MQLONG) - input
The number of possible destinations. Destinations are instances of the destination queue and are described by destination records.

- A destination record is a MQWDR structure. There is one structure for each possible route to each instance of the queue.
- MQWDR structures are addressed by an array of pointers, see DestinationArrayPtr.

DestinationChosen (MQLONG) - input/output
The chosen destination.

- The number of the MQWDR structure that identifies the route and queue instance where the message is to be sent.
- The value is in the range 1 - DestinationCount.
- On input to the exit, DestinationChosen indicates the route and queue instance that the queue manager has selected. The exit can accept this choice, or choose a different route and queue instance.

- The value set by the exit must be in the range 1 - DestinationCount. If any other value is returned, the queue manager uses the value of DestinationChosen on input to the exit.

DestinationArrayPtr (PPMQWDR) - input
The address of an array of pointers to destination records (MQWDR).

- There are DestinationCount destination records.

QArrayPtr (PPMQWQR) - input
The address of an array of pointers to queue records (MQWQR).

- If queue records are available, there are DestinationCount of them.
- If no queue records are available, QArrayPtr is the null pointer.

  Note: QArrayPtr can be the null pointer even when DestinationCount is greater than zero.

CacheContext (MQPTR) : Version 2 - input
The CacheContext field is reserved for use by the queue manager. The exit must not alter the value of this field.

CacheType (MQLONG) : Version 2 - input
The cluster cache has one of the following types:

MQCLCT_STATIC
The cache is static.

- The size of the cache is fixed, and cannot grow as the queue manager operates.
- You do not need to use the MQXCLWLN call to navigate the records in this type of cache.

MQCLCT_DYNAMIC
The cache is dynamic.

- The size of the cache can increase in order to accommodate the varying cluster information.
- You must use the MQXCLWLN call to navigate the records in this type of cache.

CLWLMRUChannels (MQLONG) : Version 3 - input
Indicates the maximum number of active outbound cluster channels, to be considered for use by the cluster workload choice algorithm.

- CLWLMRUChannels is a value 1 - 999 999 999.

pEntryPoints (PMQIEP) : Version 4
The address of an MQIEP structure through which MQI and DCI calls can be made.

Initial values and language declarations for MQWXP

Initial values and C and High Level Assembler Language declarations for MQWXP - Cluster workload exit parameter structure.

Table 25. Initial values of fields in MQWXP

Field name	Name of constant	Value of constant
StrucId	MQWXP_STRUC_ID	'WXP? '
Version	MQWXP_VERSION_2	2
ExitId	None	0
ExitReason	MQXCC_OK	0
ExitResponse	None	0
ExitResponse2	None	0
Flags	None	0
ExitUserArea	{MQXUA_NONE_ARRAY}	0

Table 25. Initial values of fields in MQWXP (continued)		
Field name	**Name of constant**	**Value of constant**
ExitData	None	""
MsgDescPtr	None	NULL
MsgBufferPtr	None	NULL
MsgBufferLength	None	0
MsgBufferPtr	None	0
QName	None	""
QMgrName	None	""
DestinationCount	None	0
DestinationChosen	None	0
DestinationArrayPtr	None	NULL
QArrayPtr	None	NULL
CacheContext	None	NULL
CacheType	MQCLCT_DYNAMIC	1
CLWLMRUChannels	None	0
pEntryPoints	None	NULL

Notes:

1. The symbol ? represents a single blank character.
2. In the C programming language, the macro variable MQWXP_DEFAULT contains the default values. Use it in the following way to provide initial values for the fields in the structure:

```
MQWDR MyWXP = {MQWXP_DEFAULT};
```

C declaration

```
typedef struct tagMQWXP {
  MQCHAR4   StrucId;              /* Structure identifier */
  MQLONG    Version;              /* Structure version number */
  MQLONG    ExitId;               /* Type of exit */
  MQLONG    ExitReason;           /* Reason for invoking exit */
  MQLONG    ExitResponse;         /* Response from exit */
  MQLONG    ExitResponse2;        /* Reserved */
  MQLONG    Feedback;             /* Reserved */
  MQLONG    Flags;                /* Flags */
  MQBYTE16  ExitUserArea;         /* Exit user area */
  MQCHAR32  ExitData;             /* Exit data */
  PMQMD     MsgDescPtr;           /* Address of message descriptor */
  PMQVOID   MsgBufferPtr;         /* Address of buffer containing some
                                     or all of the message data */
  MQLONG    MsgBufferLength;      /* Length of buffer containing message
                                     data */
  MQLONG    MsgLength;            /* Length of complete message */
  MQCHAR48  QName;                /* Queue name */
  MQCHAR48  QMgrName;             /* Name of local queue manager */
  MQLONG    DestinationCount;     /* Number of possible destinations */
  MQLONG    DestinationChosen;    /* Destination chosen */
  PPMQWDR   DestinationArrayPtr;  /* Address of an array of pointers to
                                     destination records */
  PPMQWQR   QArrayPtr;            /* Address of an array of pointers to
                                     queue records */
  /* version 1 */
```

```
  MQPTR     CacheContext;         /* Context information */
  MQLONG    CacheType;            /* Type of cluster cache */
  /* version 2 */
  MQLONG    CLWLMRUChannnels;     /* Maximum number of most recently
                                     used cluster channels */
  /* version 3 */
  PMQIEP    pEntryPoints;         /* Address of the MQIEP structure */
  /* version 4 */
};
```

High Level Assembler

```
MQWXP                          DSECT
MQWXP_STRUCID                  DS   CL4      Structure identifier
MQWXP_VERSION                  DS   F        Structure version number
MQWXP_EXITID                   DS   F        Type of exit
MQWXP_EXITREASON               DS   F        Reason for invoking exit
MQWXP_EXITRESPONSE             DS   F        Response from exit
MQWXP_EXITRESPONSE2            DS   F        Reserved
MQWXP_FEEDBACK                 DS   F        Reserved
MQWXP_RESERVED                 DS   F        Reserved
MQWXP_EXITUSERAREA             DS   XL16     Exit user area
MQWXP_EXITDATA                 DS   CL32     Exit data
MQWXP_MSGDESCPTR               DS   F        Address of message
*                                           descriptor
MQWXP_MSGBUFFERPTR             DS   F        Address of buffer containing
*                                           some or all of the message
*                                           data
MQWXP_MSGBUFFERLENGTH          DS   F        Length of buffer containing
*                                           message data
MQWXP_MSGLENGTH                DS   F        Length of complete message
MQWXP_QNAME                    DS   CL48     Queue name
MQWXP_QMGRNAME                 DS   CL48     Name of local queue manager
MQWXP_DESTINATIONCOUNT         DS   F        Number of possible
*                                           destinations
MQWXP_DESTINATIONCHOSEN        DS   F        Destination chosen
MQWXP_DESTINATIONARRAYPTR      DS   F        Address of an array of
*                                           pointers to destination
*                                           records
MQWXP_QARRAYPTR                DS   F        Address of an array of
*                                           pointers to queue records
MQWXP_CACHECONTEXT             DS   F        Context information
MQWXP_CACHETYPE                DS   F        Type of cluster cache
MQWXP_CLWLMRUCHANNELS          DS   F        Number of most recently used
*                                           channels for workload balancing

MQWXP_LENGTH                   EQU  *-MQWXP  Length of structure
                               ORG  MQWXP
MQWXP_AREA                     DS   CL(MQWXP_LENGTH)
```

MQWDR - Cluster workload destination record structure

The following table summarizes the fields in the MQWDR - Cluster workload destination record structure.

Table 26. Fields in MQWDR

Field	Description	Page
StrucId	Structure identifier	StrucId
Version	Structure version number	Version
StrucLength	Length of MQWDR structure	StrucLength
QMgrFlags	Queue-manager flags	QMgrFlags
QMgrIdentifier	Queue-manager identifier	QMgrIdentifier
QMgrName	Queue-manager name	QMgrName
ClusterRecOffset	Logical offset of first cluster record (MQWCR)	ClusterRecOffset
ChannelState	Channel state	ChannelState

Table 26. Fields in MQWDR (continued)		
Field	**Description**	**Page**
ChannelDefOffset	Logical offset of channel-definition structure (MQCD)	ChannelDefOffset
Note: The remaining fields are ignored if Version is less than MQWDR_VERSION_2.		
DestSeqNumber	Channel destination sequence number	DestSeqNumber
DestSeqFactor	Channel destination sequence factor for weighting	DestSeqFactor

The cluster workload destination record structure contains information relating to one of the possible destinations for the message. There is one cluster workload destination record structure for each instance of the destination queue.

The cluster workload destination record structure is supported in all environments.

Additionally, the MQWDR1 and MQWDR2 structures are available for backwards compatibility.

Related reference

MQ_CLUSTER_WORKLOAD_EXIT - Call description
The cluster workload exit is called by the queue manager to route a message to an available queue manager.

MQXCLWLN - Navigate Cluster workload records
The MQXCLWLN call is used to navigate through the chains of MQWDR, MQWQR, and MQWCR records stored in the cluster cache.

MQWXP - Cluster workload exit parameter structure
The following table summarizes the fields in the MQWXP - Cluster workload exit parameter structure.

MQWQR - Cluster workload queue record structure
The following table summarizes the fields in the MQWQR - Cluster workload queue record structure.

MQWCR - Cluster workload cluster record structure
The following table summarizes the fields in the MQWCR cluster workload record structure.

Fields in MQWDR - Cluster workload destination record structure

Description of the parameters in the MQWDR - Cluster workload destination record structure.

StrucId (MQCHAR4) - input
The structure identifier for the cluster workload destination record structure.

- The StrucId value is MQWDR_STRUC_ID.
- For the C programming language, the constant MQWDR_STRUC_ID_ARRAY is also defined. It has the same value as MQWDR_STRUC_ID. It is an array of characters instead of a string.

Version (MQLONG) - input
The structure version number. Version takes one of the following values:

MQWDR_VERSION_1
Version-1 cluster workload destination record.

MQWDR_VERSION_2
Version-2 cluster workload destination record.

MQWDR_CURRENT_VERSION
Current version of cluster workload destination record.

StrucLength (MQLONG) - input
The length of MQWDR structure. StrucLength takes one of the following values:

MQWDR_LENGTH_1
Length of version-1 cluster workload destination record.

MQWDR_LENGTH_2
Length of version-2 cluster workload destination record.

MQWDR_CURRENT_LENGTH
Length of current version of cluster workload destination record.

QMgrFlags (MQLONG) - input
Queue manager flags indicating properties of the queue manager that hosts the instance of the destination queue described by the MQWDR structure. The following flags are defined:

MQQMF_REPOSITORY_Q_MGR
Destination is a full repository queue manager.

MQQMF_CLUSSDR_USER_DEFINED
Cluster-sender channel was defined manually.

MQQMF_CLUSSDR_AUTO_DEFINED
Cluster-sender channel was defined automatically.

MQQMF_AVAILABLE
Destination queue manager is available to receive messages.

Other values
Other flags in the field might be set by the queue manager for internal purposes.

QMgrIdentifier (MQCHAR48) - input
The queue manager identifier is a unique identifier for the queue manager that hosts the instance of the destination queue described by the MQWDR structure.

- The identifier is generated by the queue manager.
- The length of QMgrIdentifier is MQ_Q_MGR_IDENTIFIER_LENGTH.

QMgrName (MQCHAR48) - input
The name of the queue manager that hosts the instance of the destination queue described by the MQWDR structure.

- QMgrName can be the name of the local queue manager, as well another queue manager in the cluster.
- The length of QMgrName is MQ_Q_MGR_NAME_LENGTH.

ClusterRecOffset (MQLONG) - input
The logical offset of the first MQWCR structure that belongs to the MQWDR structure.

- For static caches, ClusterRecOffset is the offset of the first MQWCR structure that belongs to the MQWDR structure.
- The offset is measured in bytes from the start of the MQWDR structure.
- Do not use the logical offset for pointer arithmetic with dynamic caches. To obtain the address of the next record, the MQXCLWLN call must be used.

ChannelState (MQLONG) - input
The state of the channel that links the local queue manager to the queue manager identified by the MQWDR structure. The following values are possible:

MQCHS_BINDING
Channel is negotiating with the partner.

MQCHS_INACTIVE
Channel is not active.

MQCHS_INITIALIZING
Channel is initializing.

MQCHS_PAUSED
Channel has paused.

MQCHS_REQUESTING
Requester channel is requesting connection.

MQCHS_RETRYING
Channel is reattempting to establish connection.

MQCHS_RUNNING
Channel is transferring or waiting for messages.

MQCHS_STARTING
Channel is waiting to become active.

MQCHS_STOPPING
Channel is stopping.

MQCHS_STOPPED
Channel has stopped.

ChannelDefOffset (MQLONG) - input
The logical offset of the channel definition (MQCD) for the channel that links the local queue manager to the queue manager identified by the MQWDR structure.

- ChannelDefOffset is like ClusterRecOffset
- The logical offset cannot be used in pointer arithmetic. To obtain the address of the next record, the MQXCLWLN call must be used.

DestSeqFactor (MQLONG) - input
The destination sequence factor that allows a choice of the channel based on weight.

- DestSeqFactor is used before the queue manager changes it.
- The workload manager increases DestSeqFactor in a way that ensures messages are distributed down channels according to their weight.

DestSeqNumber (MQLONG) - input
The cluster channel destination value before the queue manager changes it.

- The workload manager increases DestSeqNumber every time a message is put down that channel.
- Workload exits can use DestSeqNumber to decide which channel to put a message down.

Initial values and language declarations for MQWDR

Initial values and C and High Level Assembler Language declarations for MQWDR - Cluster workload destination record.

Table 27. Initial values of fields in MQWDR

Field name	Name of constant	Value of constant
StrucId	MQWDR_STRUC_ID	'WDR? '
Version	MQWDR_VERSION_1	1
StrucLength	MQWDR_CURRENT_LENGTH[3]	136
QMgrFlags	MQWDR_NONE	0
QMgrIdentifier	None	""
QMgrName	None	""
ClusterRecOffset	None	0
ChannelState	None	0
ChannelDefOffset	None	0
DestSeqNumber	None	0
DestSeqFactor	None	0

Table 27. Initial values of fields in MQWDR (continued)		
Field name	**Name of constant**	**Value of constant**

Notes:

1. The symbol ? represents a single blank character.
2. In the C programming language, the macro variable MQWDR_DEFAULT contains the default values. Use it in the following way to provide initial values for the fields in the structure:

   ```
   MQWDR MyWDR = {MQWDR_DEFAULT};
   ```

3. The initial values intentionally set the length of the structure to the length of the current version, and not version 1 of the structure.

High Level Assembler

```
MQWDR                       DSECT
MQWDR_STRUCID               DS   CL4      Structure identifier
MQWDR_VERSION               DS   F        Structure version number
MQWDR_STRUCLENGTH           DS   F        Length of MQWDR structure
MQWDR_QMGRFLAGS             DS   F        Queue-manager flags
MQWDR_QMGRIDENTIFIER        DS   CL48     Queue-manager identifier
MQWDR_QMGRNAME              DS   CL48     Queue-manager name
MQWDR_CLUSTERRECOFFSET      DS   F        Offset of first cluster
*                                         record
MQWDR_CHANNELSTATE          DS   F        Channel state
MQWDR_CHANNELDEFOFFSET      DS   F        Offset of channel definition
*                                         structure
MQWDR_LENGTH                EQU  *-MQWDR  Length of structure
                            ORG  MQWDR
MQWDR_AREA                  DS   CL(MQWDR_LENGTH)
```

C declaration

```
typedef struct tagMQWDR {
  MQCHAR4   StrucId;           /* Structure identifier */
  MQLONG    Version;           /* Structure version number */
  MQLONG    StrucLength;       /* Length of MQWDR structure */
  MQLONG    QMgrFlags;         /* Queue-manager flags */
  MQCHAR48  QMgrIdentifier;    /* Queue-manager identifier */
  MQCHAR48  QMgrName;          /* Queue-manager name */
  MQLONG    ClusterRecOffset;  /* Offset of first cluster record */
  MQLONG    ChannelState;      /* Channel state */
  MQLONG    ChannelDefOffset;  /* Offset of channel definition structure */
  /* Ver:1 */
  MQLONG    DestSeqNumber;     /* Cluster channel destination sequence number */
  MQINT64   DestSeqFactor;     /* Cluster channel factor sequence number */
  /* Ver:2 */
};
```

MQWQR - Cluster workload queue record structure

The following table summarizes the fields in the MQWQR - Cluster workload queue record structure.

Table 28. Fields in MQWQR		
Field	**Description**	**Page**
StrucId	Structure identifier	StrucId
Version	Structure version number	Version
StrucLength	Length of MQWQR structure	StrucLength
QFlags	Queue flags	QFlags
QName	Queue name	QName

Table 28. Fields in MQWQR (continued)		
Field	**Description**	**Page**
QMgrIdentifier	Queue-manager identifier	QMgrIdentifier
ClusterRecOffset	Offset of first cluster record (MQWCR)	ClusterRecOffset
QType	Queue type	QType
QDesc	Queue description	QDesc
DefBind	Default binding	DefBind
DefPersistence	Default message persistence	DefPersistence
DefPriority	Default message priority	DefPriority
InhibitPut	Whether put operations on the queue are allowed	InhibitPut
Note: The remaining fields are ignored if Version is less than MQWQR_VERSION_2.		
CWLQueuePriority	A value 0 - 9 representing the priority of the queue	CLWLQueuePriority
CLWLQueueRank	A value 0 - 9 representing the rank of the queue	CLWLQueueRank
Note: The remaining fields are ignored if Version is less than MQWQR_VERSION_3.		
DefPutResponse	Default put response	DefPutResponse

The cluster workload queue record structure contains information relating to one of the possible destinations for the message. There is one cluster workload queue record structure for each instance of the destination queue.

The cluster workload queue record structure is supported in all environments.

Additionally, the MQWQR1 and MQWQR2 structures are available for backwards compatibility.

Related reference

MQ_CLUSTER_WORKLOAD_EXIT - Call description
The cluster workload exit is called by the queue manager to route a message to an available queue manager.

MQXCLWLN - Navigate Cluster workload records
The MQXCLWLN call is used to navigate through the chains of MQWDR, MQWQR, and MQWCR records stored in the cluster cache.

MQWXP - Cluster workload exit parameter structure
The following table summarizes the fields in the MQWXP - Cluster workload exit parameter structure.

MQWDR - Cluster workload destination record structure
The following table summarizes the fields in the MQWDR - Cluster workload destination record structure.

MQWCR - Cluster workload cluster record structure
The following table summarizes the fields in the MQWCR cluster workload record structure.

Fields in MQWQR - Cluster workload queue record structure

Description of the fields in the MQWQR - Cluster workload queue record structure.

StrucId (MQCHAR4) - input
The structure identifier for the cluster workload queue record structure.

- The StrucId value is MQWQR_STRUC_ID.
- For the C programming language, the constant MQWQR_STRUC_ID_ARRAY is also defined. It has the same value as MQWQR_STRUC_ID. It is an array of characters instead of a string.

Version (MQLONG) - input
The structure version number. Version takes one of the following values:

MQWQR_VERSION_1
Version-1 cluster workload queue record.

MQWQR_VERSION_2
Version-2 cluster workload queue record.

MQWQR_VERSION_3
Version-3 cluster workload queue record.

MQWQR_CURRENT_VERSION
Current version of cluster workload queue record.

StrucLength (MQLONG) - input
The length of MQWQR structure. StrucLength takes one of the following values:

MQWQR_LENGTH_1
Length of version-1 cluster workload queue record.

MQWQR_LENGTH_2
Length of version-2 cluster workload queue record.

MQWQR_LENGTH_3
Length of version-3 cluster workload queue record.

MQWQR_CURRENT_LENGTH
Length of current version of cluster workload queue record.

QFlags (MQLONG) - input
The queue flags indicate properties of the queue. The following flags are defined:

MQQF_LOCAL_Q
Destination is a local queue.

MQQF_CLWL_USEQ_ANY
Allow use of local and remote queues in puts.

MQQF_CLWL_USEQ_LOCAL
Allow only local queue puts.

Other values
Other flags in the field might be set by the queue manager for internal purposes.

QName (MQCHAR48) - input
The name of the queue that is one of the possible destinations of the message.

- The length of QName is MQ_Q_NAME_LENGTH.

QMgrIdentifier (MQCHAR48) - input
The queue manager identifier is a unique identifier for the queue manager that hosts the instance of the queue described by the MQWQR structure.

- The identifier is generated by the queue manager.
- The length of QMgrIdentifier is MQ_Q_MGR_IDENTIFIER_LENGTH.

ClusterRecOffset (MQLONG) - input
The logical offset of the first MQWCR structure that belongs to the MQWQR structure.

- For static caches, ClusterRecOffset is the offset of the first MQWCR structure that belongs to the MQWQR structure.
- The offset is measured in bytes from the start of the MQWQR structure.
- Do not use the logical offset for pointer arithmetic with dynamic caches. To obtain the address of the next record, the MQXCLWLN call must be used.

QType (MQLONG) - input
The queue type of the destination queue. The following values are possible:

MQCQT_LOCAL_Q
Local queue.

MQCQT_ALIAS_Q
Alias queue.

MQCQT_REMOTE_Q
Remote queue.

MQCQT_Q_MGR_ALIAS
Queue-manager alias.

QDesc (MQCHAR64) - input
The queue description queue attribute defined on the queue manager that hosts the instance of the destination queue described by the MQWQR structure.

- The length of QDesc is MQ_Q_DESC_LENGTH.

DefBind (MQLONG) - input
The default binding queue attribute defined on the queue manager that hosts the instance of the destination queue described by the MQWQR structure. Either MQBND_BIND_ON_OPEN or MQBND_BIND_ON_GROUP must be specified when using groups with clusters. The following values are possible:

MQBND_BIND_ON_OPEN
Binding fixed by MQOPEN call.

MQBND_BIND_NOT_FIXED
Binding not fixed.

MQBND_BIND_ON_GROUP
Allows an application to request that a group of messages are all allocated to the same destination instance.

DefPersistence (MQLONG) - input
The default message persistence queue attribute defined on the queue manager that hosts the instance of the destination queue described by the MQWQR structure. The following values are possible:

MQPER_PERSISTENT
Message is persistent.

MQPER_NOT_PERSISTENT
Message is not persistent.

DefPriority (MQLONG) - input

The default message priority queue attribute defined on the queue manager that hosts the instance of the destination queue described by the MQWQR structure. The priority range is 0 - MaxPriority.

- 0 is the lowest priority.
- MaxPriority is the queue manager attribute of the queue manager that hosts this instance of the destination queue.

InhibitPut (MQLONG) - input
The put inhibited queue attribute defined on the queue manager that hosts the instance of the destination queue described by the MQWQR structure. The following values are possible:

MQQA_PUT_INHIBITED
Put operations are inhibited.

MQQA_PUT_ALLOWED
Put operations are allowed.

CLWLQueuePriority (MQLONG) - input
The cluster workload queue priority attribute defined on the queue manager that hosts the instance of the destination queue described by the MQWQR structure.

CLWLQueueRank (MQLONG) - input
The cluster workload queue rank defined on the queue manager that hosts the instance of the destination queue described by the MQWQR structure.

DefPutResponse (MQLONG) - input
The default put response queue attribute defined on the queue manager that hosts the instance of the destination queue described by the MQWQR structure. The following values are possible:

MQPRT_SYNC_RESPONSE
Synchronous response to MQPUT or MQPUT1 calls.

MQPRT_ASYNC_RESPONSE
Asynchronous response to MQPUT or MQPUT1 calls.

Initial values and language declarations for MQWQR

Initial values and C and High Level Assembler Language declarations for MQWQR - Cluster workload queue record.

Table 29. Initial values of fields in MQWQR

Field name	Name of constant	Value of constant
StrucId	MQWQR_STRUC_ID_ARRAY	'WQR? '
Version	MQWQR_VERSION_1	1
StrucLength	MQWQR_CURRENT_LENGTH[3]	212
QFlags	None	0
QName	None	""
QMgrIdentifier	None	""
ClusterRecOffset	None	0
QType	None	0
QDesc	None	""
DefBind	None	0
DefPersistence	None	0
DefPriority	None	0
InhibitPut	None	0
CLWLQueuePriority	None	0
CLWLQueueRank	None	0
DefPutResponse	None	1

Notes:

1. The symbol ? represents a single blank character.
2. In the C programming language, the macro variable MQWQR_DEFAULT contains the default values. Use it in the following way to provide initial values for the fields in the structure:

```
MQWQR MyWQR = {MQWQR_DEFAULT};
```

3. The initial values intentionally set the length of the structure to the length of the current version, and not version 1 of the structure.

C declaration

```
typedef struct tagMQWQR {
  MQCHAR4   StrucId;           /* Structure identifier */
  MQLONG    Version;           /* Structure version number */
  MQLONG    StrucLength;       /* Length of MQWQR structure */
  MQLONG    QFlags;            /* Queue flags */
  MQCHAR48  QName;             /* Queue name */
  MQCHAR48  QMgrIdentifier;    /* Queue-manager identifier */
  MQLONG    ClusterRecOffset;  /* Offset of first cluster record */
  MQLONG    QType;             /* Queue type */
  MQCHAR64  QDesc;             /* Queue description */
  MQLONG    DefBind;           /* Default binding */
  MQLONG    DefPersistence;    /* Default message persistence */
  MQLONG    DefPriority;       /* Default message priority */
  MQLONG    InhibitPut;        /* Whether put operations on the queue
                                  are allowed */
  /* version 2 */
  MQLONG    CLWLQueuePriority; /* Queue priority */
  MQLONG    CLWLQueueRank;     /* Queue rank */
  /* version 3 */
  MQLONG    DefPutResponse;    /* Default put response */
};
```

High Level Assembler

```
MQWQR                          DSECT
MQWQR_STRUCID                  DS   CL4      Structure identifier
MQWQR_VERSION                  DS   F        Structure version number
MQWQR_STRUCLENGTH              DS   F        Length of MQWQR structure
MQWQR_QFLAGS                   DS   F        Queue flags
MQWQR_QNAME                    DS   CL48     Queue name
MQWQR_QMGRIDENTIFIER           DS   CL48     Queue-manager identifier
MQWQR_CLUSTERRECOFFSET         DS   F        Offset of first cluster
*                                           record
MQWQR_QTYPE                    DS   F        Queue type
MQWQR_QDESC                    DS   CL64     Queue description
MQWQR_DEFBIND                  DS   F        Default binding
MQWQR_DEFPERSISTENCE           DS   F        Default message persistence
MQWQR_DEFPRIORITY              DS   F        Default message priority
MQWQR_INHIBITPUT               DS   F        Whether put operations on
*                                           the queue are allowed
MQWQR_DEFPUTRESPONSE           DS   F        Default put response
MQWQR_LENGTH                   EQU  *-MQWQR  Length of structure
                               ORG  MQWQR
MQWQR_AREA                     DS   CL(MQWQR_LENGTH)
```

MQWCR - Cluster workload cluster record structure

The following table summarizes the fields in the MQWCR cluster workload record structure.

Table 30. Fields in MQWCR

Field	Description	Page
ClusterName	Name of cluster	ClusterName
ClusterRecOffset	Offset of next cluster record (MQWCR)	ClusterRecOffset
ClusterFlags	Cluster flags	ClusterFlags

The cluster workload cluster record structure contains information about a cluster. For each cluster the destination queue belongs to, there is one cluster workload cluster record structure.

The cluster workload cluster record structure is supported in all environments.

Related reference

MQ_CLUSTER_WORKLOAD_EXIT - Call description
The cluster workload exit is called by the queue manager to route a message to an available queue manager.

MQXCLWLN - Navigate Cluster workload records

The MQXCLWLN call is used to navigate through the chains of MQWDR, MQWQR, and MQWCR records stored in the cluster cache.

MQWXP - Cluster workload exit parameter structure
The following table summarizes the fields in the MQWXP - Cluster workload exit parameter structure.

MQWDR - Cluster workload destination record structure
The following table summarizes the fields in the MQWDR - Cluster workload destination record structure.

MQWQR - Cluster workload queue record structure
The following table summarizes the fields in the MQWQR - Cluster workload queue record structure.

Fields in the MQWCR - Cluster workload cluster record structure.

Description of the fields in the MQWCR - Cluster workload cluster record structure.

ClusterName (MQCHAR48) - input
The name of a cluster to which the instance of the destination queue that owns the MQWCR structure belongs. The destination queue instance is described by an MQWDR structure.

- The length of ClusterName is MQ_CLUSTER_NAME_LENGTH.

ClusterRecOffset (MQLONG) - input
The logical offset of the next MQWCR structure.

- If there are no more MQWCR structures, ClusterRecOffset is zero.
- The offset is measured in bytes from the start of the MQWCR structure.

ClusterFlags (MQLONG) - input
The cluster flags indicate properties of the queue manager identified by the MQWCR structure. The following flags are defined:

MQQMF_REPOSITORY_Q_MGR
Destination is a full repository queue manager.

MQQMF_CLUSSDR_USER_DEFINED
Cluster-sender channel was defined manually.

MQQMF_CLUSSDR_AUTO_DEFINED
Cluster-sender channel was defined automatically.

MQQMF_AVAILABLE
Destination queue manager is available to receive messages.

Other values
Other flags in the field might be set by the queue manager for internal purposes.

Related reference
Initial values and language declarations for MQWCR
Initial values and C and High Level Assembler Language declarations for MQWCR - Cluster workload cluster record structure.

Initial values and language declarations for MQWCR

Initial values and C and High Level Assembler Language declarations for MQWCR - Cluster workload cluster record structure.

Table 31. Initial values of fields in MQWCR

Field name	Name of constant	Value of constant
ClusterName	None	" "
ClusterRecOffset	None	0
ClusterFlags	None	0

C declaration

```
typedef struct tagMQWCR {
  MQCHAR48  ClusterName;       /* Cluster name */
  MQLONG    ClusterRecOffset;  /* Offset of next cluster record */
  MQLONG    ClusterFlags;      /* Cluster flags */
};
```

High Level Assembler

```
MQWCR                           DSECT
MQWCR_CLUSTERNAME               DS   CL48     Cluster name
MQWCR_CLUSTERRECOFFSET          DS   F        Offset of next cluster
*                                            record
MQWCR_CLUSTERFLAGS              DS   F        Cluster flags
MQWCR_LENGTH                    EQU  *-MQWCR  Length of structure
                                ORG  MQWCR
MQWCR_AREA                      DS   CL(MQWCR_LENGTH)
```

Related reference

Fields in the MQWCR - Cluster workload cluster record structure.
Description of the fields in the MQWCR - Cluster workload cluster record structure.

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 32. 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 WebSphere MQ listener), runmqchl (Run WebSphere MQ channel), and runmqchi (Run WebSphere 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

Environment variables

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

Examples of use

- On UNIX and Linux systems use: `export [environment variable]=filename`.
- On Windows Systems, use: `Set [environment variable]=filename`.
-

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

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

AMQ_SSL_ALLOW_DEFAULT_CERT
When the `AMQ_SSL_ALLOW_DEFAULT_CERT` environment variable is not set, an application can connect to a queue manager with a personal certificate in the client keystore only when the certificate includes the label name of `ibmwebspheremq<userid>`. When the `AMQ_SSL_ALLOW_DEFAULT_CERT` environment variable is set, the certificate does not require the label name of `ibmwebspheremq<userid>`. That is, the certificate that is used to connect to a queue manager can be a default certificate, provided that a default certificate is present in the key repository, and the key repository does not contain a personal certificate with the prefix `ibmwebspheremq<userid>`. For more information, see the technote Specifying the userid in the SSL certificate label for an MQ client.

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

V 7.5.0.9

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

Set the variable to appropriate value in the environment that is used to start the queue manager or channel. To request that LDAP v2 is used, set `AMQ_SSL_LDAP_SERVER_VERSION=2`. To request that LDAP v3 is used, set `AMQ_SSL_LDAP_SERVER_VERSION=3`.

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

GMQ_MQ_LIB
When both the IBM WebSphere MQ MQI client and IBM WebSphere MQ server are installed on your system, MQAX applications run against the server by default. To run MQAX against the client, the client bindings library must be specified in the GMQ_MQ_LIB environment variable, for example, set `GMQ_MQ_LIB=mqic.dll`. For a client only installation, it is not necessary to set the GMQ_MQ_LIB environment variable. When this variable is not set, WebSphere MQ attempts to load `amqzst.dll`. If this DLL is not present (as is the case in a client only installation), WebSphere MQ attempts to load `mqic.dll`.

HOME
This variable contains the name of the directory which is searched for the `mqclient.ini` file. This file contains configuration information used by IBM WebSphere MQ MQI clients on UNIX and Linux systems.

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

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

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

LDAP_VERSION
An optional variable for running an LDAP sample program. It specifies the version of the LDAP protocol to be used, and can be either 2 or 3. Most LDAP servers now support version 3 of the

protocol; they all support the older version 2. This sample works equally well with either version of the protocol, and if it is not specified it defaults to version 2.

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

MQCCSID
Specifies the coded character set number to be used and overrides the native CCSID of the application.

MQCERTVPOL
Determines the type of certificate validation used:

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

RFC5280
Use only certificate validation which complies with the RFC 5280 standard.

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

MQCHLTAB
MQCHLTAB specifies the name of the file containing the client channel definition table (ccdt). The default file name is AMQCLCHL.TAB.

MQC_IPC_HOST
When sharing IBM WebSphere MQ files and the generated value of `myHostName` creates a problem set `myHostName` using the environment variable MQC_IPC_HOST

MQCLNTCF
Use this environment variable to modify the `mqclient.ini` file path.

MQ_CHANNEL_SUPPRESS_INTERVAL
Specifies the time interval, in seconds, during which the messages defined with MQ_CHANNEL_SUPPRESS_MSGS are to be suppressed from being written to the error log, together with the number of times that a message will be allowed to occur during the specified time interval before being suppressed. The default value is 60,5 which means any further occurrences of a given message are suppressed after the first five occurrences of that message in a 60 second interval. For more information, see Suppressing channel error messages from error logs.

The environment variable MQ_CHANNEL_SUPPRESS_INTERVAL is comparable to SuppressInterval in the `qm.ini` file.

MQ_CHANNEL_SUPPRESS_MSGS
Specifies IBM WebSphere MQ channel error messages that are to be written to the error log only for the specified number of times that these messages are allowed to occur during the time interval defined in MQ_CHANNEL_SUPPRESS_INTERVAL before being suppressed until that time interval expires. For more information, see Suppressing channel error messages from error logs.

The environment variable MQ_CHANNEL_SUPPRESS_MSGS is comparable to SuppressMessage in the `qm.ini` file, although it is specified differently.

MQ_CONNECT_TYPE
On IBM WebSphere MQ for Windows, UNIX and Linux systems, use this environment variable in combination with the type of binding specified in the Options field of the MQCNO structure used on an MQCONNX call. See MQCONNX environment variable

MQ_FILE_PATH
During the installation of the runtime package on the Windows platform, a new environment variable called MQ_FILE_PATH is configured. This environment variable contains the same data as the following key in the Windows Registry:

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

MQIPADDRV
MQIPADDRV specifies which IP protocol to use for a channel connection. It has the possible string values of "MQIPADDR_IPV4" or "MQIPADDR_IPV6". These values have the same meanings as IPV4 and IPV6 in ALTER QMGR IPADDRV. If it is not set, "MQIPADDR_IPV4" is assumed.

MQ_JAVA_DATA_PATH
Specifies the directory for log and trace output.

MQ_JAVA_INSTALL_PATH
Specifies the directory where IBM WebSphere MQ classes for Java are installed, as shown in IBM WebSphere MQ classes for Java installation directories.

MQ_JAVA_LIB_PATH
Specifies the directory where the IBM WebSphere MQ classes for Java libraries are stored. Some scripts supplied with IBM WebSphere MQ classes for Java, such as IVTRun, use this environment variable.

MQNAME
MQNAME specifies the local NetBIOS name that the IBM WebSphere MQ processes can use.

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

MQPSE_TRACE_LOGFILE
Use when you Publish the Exit Sample Program. In the application process to be traced, this environment variable describes where the trace files must be written to. See The Publish Exit sample program

MQSERVER
MQSERVER environment variable is used to define a minimal channel. You cannot use MQSERVER to define an SSL channel or a channel with channel exits. MQSERVER specifies the location of the WebSphere MQ server and the communication method to be used.

MQ_SET_NODELAYACK
When you set this variable, it switches off TCP delayed acknowledgment

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

MQSNOAUT
MQSNOAUT disables the object authority manager (OAM) and prevents any security checking. The MQSNOAUT variable only takes effect when a queue manager is created.

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

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

MQSSLCRYP
MQSSLCRYP holds a parameter string that you can use to configure the cryptographic hardware present on the system. The permitted values are the same as for the SSLCRYP parameter of the ALTER QMGR command.

MQSSLFIPS
MQSSLFIPS specifies whether only FIPS-certified algorithms are to be used if cryptography is carried out in IBM WebSphere MQ. The values are the same as for the SSLFIPS parameter of the ALTER QMGR command.

MQSSLKEYR
MQSSLKEYR specifies the location of the key repository that holds the digital certificate belonging to the user, in stem format. Stem format means that it includes the full path and the file name without an extension. For full details, see the SSLKEYR parameter of the ALTER QMGR command.

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

MQSSLRESET
MQSSLRESET represents the number of unencrypted bytes sent and received on an SSL channel before the SSL secret key is renegotiated.

MQS_TRACE_OPTIONS
Use the environment variable MQS_TRACE_OPTIONS to activate the high detail and parameter tracing functions individually.

MQTCPTIMEOUT
This variable specifies how long IBM WebSphere MQ waits for a TCP connect call.

MQSUITEB
This variable specifies whether Suite B compliant cryptography is to be used. In the instance that Suite B cryptography is used you can specify the strength of the cryptography by setting MQSUITEB to one of the following:

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

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

ODQ_TRACE
If you use a dead-letter queue handler that is different from RUNMQDLQ the source of the sample is available for you to use as your base. The sample is like the dead-letter handler provided within the product but trace and error reporting are different. Set the ODQ_TRACE environment variable to YES or yes to switch on tracing

OMQ_PATH
This environment variable is where you can find the First Failure Symptom report if your IBM WebSphere MQ automation classes for ActiveX script fails.

OMQ_TRACE
MQAX includes a trace facility to help the service organization identify what is happening when you have a problem. It shows the paths taken when you run your MQAX script. Unless you have a problem, run with tracing set off to avoid any unnecessary use of system resources. OMQ_TRACE is one of the three environment variables set to control trace. Specifying any value for OMQ_TRACE switches the trace facility on. Even if you set OMQ_TRACE to OFF, trace is still active. See Using trace

OMQ_TRACE_PATH
One of the three environment variables set to control trace. See Using trace

OMQ_TRACE_LEVEL
One of the three environment variables set to control trace. See Using trace

ONCONFIG
The name of the Informix server configuration file. For example, on UNIX and Linux systems, use:

```
export ONCONFIG=onconfig.hostname_1
```

On Windows systems, use:

```
set ONCONFIG=onconfig.hostname_1
```

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

WMQSOAP_HOME
Use when making additional configuration steps after the .NET SOAP over JMS service hosting environment is correctly installed and configured in IBM WebSphere MQ. It is accessible from a local queue manager. See WCF client to a .NET service hosted by WebSphere MQ sample and WCF client to an Axis Java service hosted by WebSphere MQ sample

Also use when you install WebSphere MQ web transport for SOAP. See Installing WebSphere MQ Web transport for SOAP

Message channel planning example for distributed platforms

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 WebSphere 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.

What the example shows

The example shows the WebSphere 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 4 on page 137 shows the example scenario.

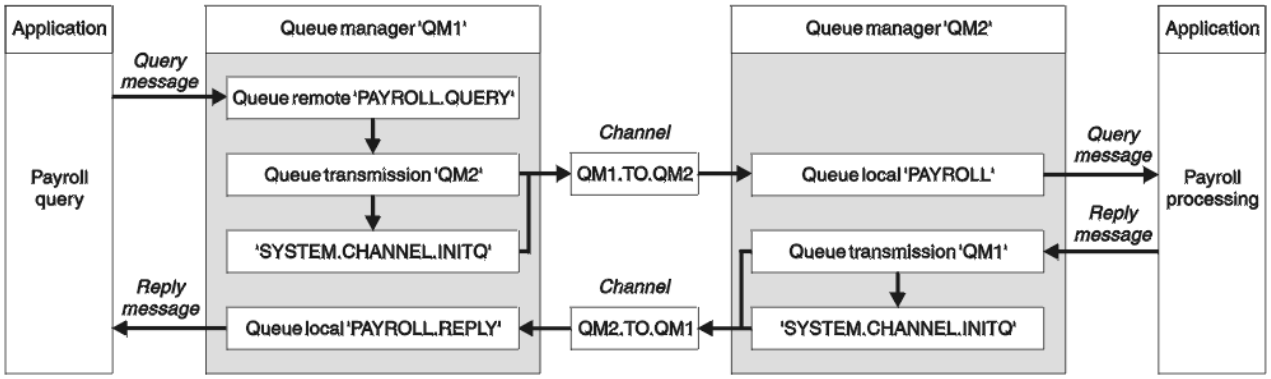

Figure 4. The message channel example for Windows, UNIX and Linux systems

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 4 on page 137.

Queue manager QM1 example

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 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.

Queue manager QM2 example

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')
```

Running the example

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.

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 *
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.

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 IBM WebSphere MQ.

However, this information may also contain diagnosis, modification, and tuning information. Diagnosis, modification and tuning information is provided to help you debug your application software.

Important: Do not use this diagnosis, modification, and tuning information as a programming interface because it is subject to change.

Trademarks

IBM, the IBM logo, ibm.com®, are trademarks of IBM Corporation, registered in many jurisdictions worldwide. A current list of IBM trademarks is available on the Web at "Copyright and trademark information"www.ibm.com/legal/copytrade.shtml. Other product and service names might be trademarks of IBM or other companies.

Microsoft and Windows are trademarks of Microsoft Corporation in the United States, other countries, or both.

UNIX is a registered trademark of The Open Group in the United States and other countries.

Linux is a registered trademark of Linus Torvalds in the United States, other countries, or both.

This product includes software developed by the Eclipse Project (http://www.eclipse.org/).

Java and all Java-based trademarks and logos are trademarks or registered trademarks of Oracle and/or its affiliates.

Part Number:

(1P) P/N: