9.3

IBM MQ Configuration Reference

Note

Before using this information and the product it supports, read the information in "Notices" on page 257.

This edition applies to version 9 release 3 of IBM® MQ and to all subsequent releases and modifications until otherwise indicated in new editions.

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

Contents

Configuration reference

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

The configuration reference information is provided in the following subtopics:

Related tasks

Configuring

z/OS Configuring z/OS

Example: setting up cross-platform communication for IBM MQ

This example shows how to establish a working IBM MQ network by configuring IBM MQ sender and receiver channels to enable bidirectional message flow between the platforms over all supported protocols.

Before you begin

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

- z/OS z/OS communicates by using a 3745 network controller (or equivalent)

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).

The example configurations cover the following network software products:

- SNA
  - Windows IBM Personal Communications for Windows
  - AIX IBM Communications Server for AIX®
  - IBM i IBM i
  - OS/390®
- TCP
  - Windows Microsoft Windows
  - AIX AIX
  - IBM i IBM i
  - z/OS TCP for z/OS
- Windows NetBIOS
- Windows SPX

For more information about supported communications protocols and software, see System Requirements for IBM MQ.

About this task

This example uses sender and receiver channels. To use channel types other than sender-receiver, DEFINE CHANNEL (define a new channel).

Figure 1 on page 6 is a conceptual representation of a single channel and the IBM MQ objects that are associated with it.

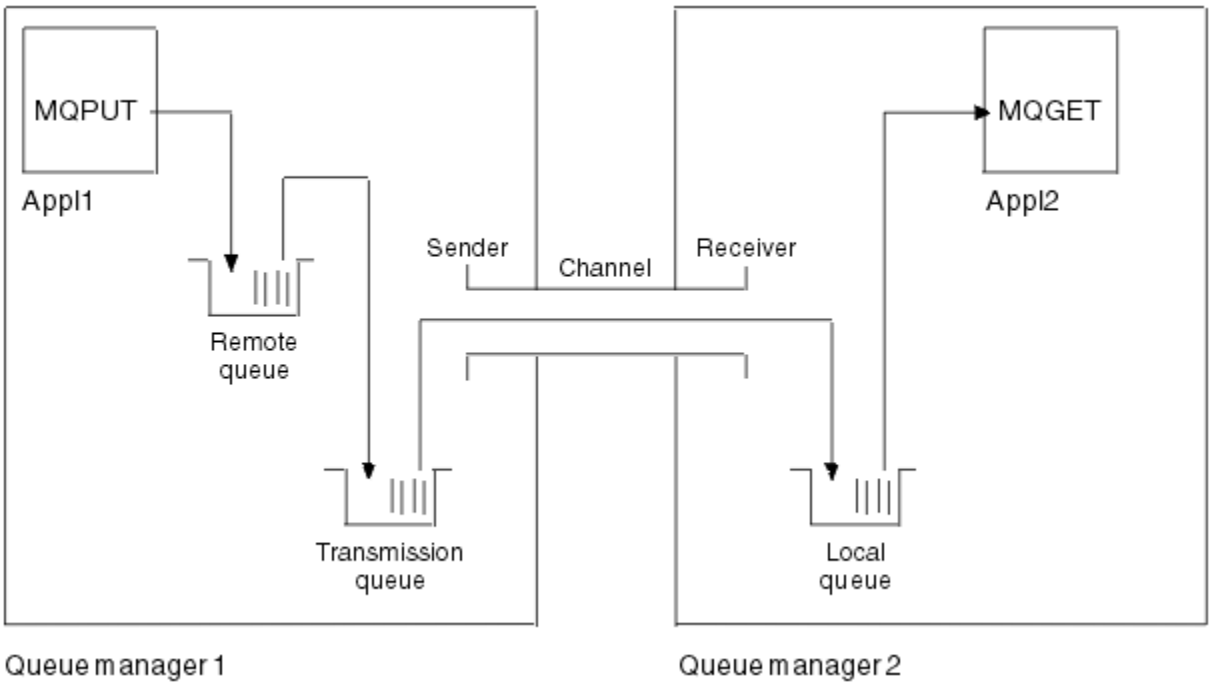

Figure 1. IBM MQ channel to be set up in the example configuration

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

The objects in this network are:

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

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

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

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

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

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

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

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

Procedure

1. Read through the information in "How to use the cross-platform communication examples" on page 7.
2. Follow the instructions for the appropriate platform or platforms to establish a network connection and define the channels.
   a) AIX
   See "Example: setting up IBM MQ cross-platform communication on AIX" on page 9
   b) IBM i
   See "Example: setting up IBM MQ cross-platform communication on IBM i" on page 15
   c) Linux
   See "Example: setting up IBM MQ cross-platform communication on Linux" on page 32
   d) Windows
   See "Example: setting up IBM MQ cross-platform communication on Windows" on page 38
   e) z/OS
   See "Example: setting up IBM MQ cross-platform communication on z/OS" on page 45
   f) z/OS
   See "Example: setting up IBM MQ cross-platform communication on z/OS using QSGs" on page 49
   g) z/OS
   See "Example: setting up IBM MQ cross-platform communication for intra-group queuing on z/OS" on page 57

Related tasks

Configuring distributed queuing

Setting up communications with other queue managers on z/OS

How to use the cross-platform communication examples

The example configurations for setting up cross-platform communication for IBM MQ describe the tasks that are carried out on a single platform to set up communication to another platform. The examples then 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.

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

- Linux AIX On AIX and Linux:

  ```
  MQ_INSTALLATION_PATH/bin/setmqenv
  ```

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

  ```
  MQ_INSTALLATION_PATH/bin/setmqenv
  ```

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

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

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

There are example configurations for the following platforms:

- AIX "Example: setting up IBM MQ cross-platform communication on AIX" on page 9
- IBM i "Example: setting up IBM MQ cross-platform communication on IBM i" on page 15
- Linux "Example: setting up IBM MQ cross-platform communication on Linux" on page 32
- Windows "Example: setting up IBM MQ cross-platform communication on Windows" on page 38
- z/OS "Example: setting up IBM MQ cross-platform communication on z/OS" on page 45
- z/OS "Example: setting up IBM MQ cross-platform communication on z/OS using QSGs" on page 49
- z/OS "Example: setting up IBM MQ cross-platform communication for intra-group queuing on z/OS" on page 57

IT responsibilities

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

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

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

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

Related tasks

"Example: setting up cross-platform communication for IBM MQ" on page 5
This example shows how to establish a working IBM MQ network by configuring IBM MQ sender and receiver channels to enable bidirectional message flow between the platforms over all supported protocols.

Related reference

setmqenv

Example: setting up IBM MQ cross-platform communication on AIX

This example shows how to set up communication links from IBM MQ on AIX to IBM MQ on another platform and establish a working channel to that platform.

Before you begin

For background information about this example and how to use it, see "Example: setting up cross-platform communication for IBM MQ" on page 5 and "How to use the cross-platform communication examples" on page 7.

About this task

This example covers setting up cross platform communication from IBM MQ on AIX to the following platforms:

- 

Windows
- 

Linux
- 

IBM i
- 

z/OS
- VSE/ESA

Procedure

1. Establish a network connection using one of the following options.
   - Establish an LU 6.2 connection. For more information about configuring SNA over TCP/IP, see Communications Server for AIX Library.
   - Establish 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. Use the following command to start the IBM MQ for TCP listener:

     ```
     runmqlsr -t tcp
     ```

     a. 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
     ```

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

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

c. 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.

2. After the connection is established, define some channels as described in "Configuring the channels on AIX" on page 10.

AIX Configuring the channels on AIX

To configure IBM MQ for the example configuration on AIX, complete the basic configuration steps for the queue manager, then configure the sender and receiver channels.

Before you begin

Note:

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

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

3. Start any channel using the command:

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

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

Procedure

1. To carry out the basic configuration for the queue manager, complete the following steps:
   a) 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.

b) 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.

c) 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.

2. Configure the channels for the example configuration.

For more information about the parameters used in the following examples, see "Channel configuration parameters for AIX" on page 12. In each case, the example shows the MQSC command. Either start **runmqsc** from an AIX command line and enter each command in turn, or build the commands into a command file.

Windows These examples are for connecting IBM MQ on AIX with IBM MQ on Windows. To connect to IBM MQ on another platform, use the appropriate values from the tables in "Channel configuration parameters for AIX" on page 12 instead of the values for Windows.

a) Define the sender channel as shown in the following examples:

- Using SNA:

```
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
```

- Using TCP:

```
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
```

b) Define the receiver channel as shown in the following examples:

- Using SNA:

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

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

- Using TCP:

```
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
```

Note: There are alternative ways of ensuring that SNA receiver channels activate correctly when a sender channel initiates a conversation.

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

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

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

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

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

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

AIX *Channel configuration parameters for AIX*

The parameters needed to configure the channels for the example configuration on AIX.

Step “2” on page 11 of “Configuring the channels on AIX” on page 10 describes the configuration to be performed on the AIX queue manager to implement the channel described in “Example: setting up cross-platform communication for IBM MQ” on page 5. The examples in “Configuring the channels on AIX” on page 10 are for connecting IBM MQ for IBM i and IBM MQ for Windows. To connect to IBM MQ on another platform, use the values from the appropriate table in place of the values for Windows.

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

Definition for local node

Table 1. Configuration examples for the definition for the local node

ID	Parameter name	Reference	Example used
A	Queue Manager Name		**AIX**
B	Local queue name		**AIX.LOCALQ**

Connection to IBM MQ on Windows

Windows

The values in this section of the table must match those used in "Channel configuration parameters for Windows" on page 42, as indicated.

Table 2. Configuration examples for connecting to IBM MQ on Windows

ID	Parameter name	Reference	Example used
C	Remote queue manager name	A	**WINNT**
D	Remote queue name		**WINNT.REMOTEQ**
E	Queue name at remote system	B	**WINNT.LOCALQ**
F	Transmission queue name		**WINNT**
G	Sender (SNA) channel name		**AIX.WINNT.SNA**
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 IBM MQ on Linux

Linux

The values in this section of the table must match those used in "Channel configuration parameters for Linux" on page 36, as indicated.

Table 3. Configuration examples for connecting to IBM MQ on Linux

ID	Parameter name	Reference	Example used
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**

Table 3. Configuration examples for connecting to IBM MQ on Linux (continued)			
ID	**Parameter name**	**Reference**	**Example used**
I	Receiver (SNA) channel name	G	**LINUX.AIX.SNA**
J	Receiver (TCP/IP) channel name	H	**LINUX.AIX.TCP**

Connection to IBM MQ on IBM i

IBM i

The values in this section of the table must match those used in "Channel configuration parameters for IBM i" on page 29, as indicated.

Table 4. Configuration examples for connecting to IBM MQ on IBM i			
ID	**Parameter Name**	**Reference**	**Example Used**
C	Remote queue manager name	A	**AS400**
D	Remote queue name		**AS400.REMOTEQ**
E	Queue name at remote system	B	**AS400.LOCALQ**
F	Transmission queue name		**AS400**
G	Sender (SNA) channel name		**AIX.AS400.SNA**
H	Sender (TCP) channel name		**AIX.AS400.TCP**
I	Receiver (SNA) channel name	G	**AS400.AIX.SNA**
J	Receiver (TCP) channel name	H	**AS400.AIX.TCP**

Connection to IBM MQ for z/OS

z/OS

The values in this section of the table must match those used in "Channel configuration parameters for z/OS" on page 47, as indicated.

Table 5. Configuration examples for connecting to IBM MQ for z/OS			
ID	**Parameter name**	**Reference**	**Example used**
C	Remote queue manager name	A	**MVS**
D	Remote queue name		**MVS.REMOTEQ**
E	Queue name at remote system	B	**MVS.LOCALQ**
F	Transmission queue name		**MVS**
G	Sender (SNA) channel name		**AIX.MVS.SNA**
H	Sender (TCP) channel name		**AIX.MVS.TCP**
I	Receiver (SNA) channel name	G	**MVS.AIX.SNA**
J	Receiver (TCP) channel name	H	**MVS.AIX.TCP**

Connection to IBM MQ for z/OS using queue sharing groups

z/OS

The values in this section of the table must match those used in "Shared channel configuration parameters" on page 56, as indicated.

Table 6. Configuration examples for connecting to IBM MQ for z/OS using queue sharing groups

ID	Parameter name	Reference	Example used
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**
J	Receiver (TCP) channel name	H	**QSG.AIX.TCP**

IBM i Example: setting up IBM MQ cross-platform communication on IBM i

This example shows how to set up communication links from IBM MQ on IBM i to IBM MQ on another platform and establish a working channel to that platform.

Before you begin

For background information about this example and how to use it, see "Example: setting up cross-platform communication for IBM MQ" on page 5 and "How to use the cross-platform communication examples" on page 7.

About this task

This example covers setting up cross platform communication from IBM MQ on IBM i to the following platforms:

- Windows Windows
- AIX AIX
- Linux Linux
- z/OS z/OS or MVS
- VSE/ESA

Procedure

1. Establish a network connection using one of the following options.
   - Establish an LU 6.2 connection as described in "Establishing an LU 6.2 connection on IBM i" on page 16.
   - Establish a TCP connection as described in "Establishing a TCP connection on IBM i" on page 24.

2. After the connection is established, define some channels as described in “Configuring the channels on IBM i” on page 26.

IBM i Establishing an LU 6.2 connection on IBM i

To establish an LU 6.2 connection on IBM i, you need to configure the local node and connect it to a partner node.

About this task

For more information about the parameters that are needed to set up communication from IBM i system to one of the other IBM MQ platforms, see the tables in “Configuration parameters for an LU 6.2 connection on IBM i” on page 20. The numbers in brackets () in the task steps correspond to the values in the *ID* column of these tables.

To configure the local node, you need to:

- Create a line description
- Add a routing entry then start the subsystem

To connect to a partner node, you need to:

- Create a controller description
- Create a device description
- Create the CPI-C side information
- Add a communications entry for APPC
- Add a configuration list entry

Procedure

1. Configure the local node by creating a line description and adding a routing entry.
   a) Create a line description.
      If the line description has not already been created, use the **CRTLINTRN** command to specify values for **Line description** (6) and **Resource name** (7) as shown in the following example:

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

Type choices, press Enter.

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



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

   b) Add a routing entry.
      Type the command **ADDRTGE** and press Enter, then specify your own value for **Subsystem description** (5), and the values that are shown in the following example for **Routing entry sequence number**, **Compare value** (8), **Starting position**, **Program to call**, and the **Library** containing the program to call.

```
Add Routing Entry (ADDRTGE)

Type choices, press Enter.

Subsystem description . . . . .  QCMN     Name
Library . . . . . .   *LIBL    Name, *LIBL, *CURLIB
Routing entry sequence number .  1       1-9999
Comparison data:
Compare value . . . . . .  'MQSERIES'

Starting position . . . . . .  37      1-80
Program to call . . . . . .  AMQCRC6B   Name, *RTGDTA
Library . . . . . .   QMAS400    Name, * LI BL, *CURLIB
Class . . . . . .  *SBSD     Name, *SBSD
Library . . . . . .  *LIBL     Name, *LIBL, *CURLIB
Maximum active routing steps . .  *NOMAX    0-1000, *NOMAX
Storage pool identifier . . . .  1       1-10



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

Start the subsystem by typing the command STRSBS *subsystem description* (5) and pressing Enter.

2. Create the connection to the partner node by creating a controller description, a device description, and the CPI-C side information, and adding a communications entry for APPC, and a configuration list entry.

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

   a) Create a controller description

   At a command-line type CRTCTLAPPC and press Enter, then specify values for **Controller description** (12), set **Link type** to *LAN, and set **Online at IPL** to *NO.

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

Type choices, press Enter.

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



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

   Press Enter twice, followed by F10, then specify values for **Switched line list** (6), **Remote network identifier** (9), **Remote control point** (10), and **LAN remote adapter address** (16) and press Enter.

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

Type choices, press Enter.

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

b) Create a device description.

Type the command CRTDEVAPPC and press Enter, then specify values for **Device description** (13), **Remote location** (11), **Local location** (3), **Remote network identifier** (9), and **Attached controller** (12).

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

Type choices, press Enter.

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


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

Note: You can avoid having to create controller and device descriptions manually by taking advantage of the IBM i auto-configuration service. For more information, see the IBM i documentation.

c) Create the CPI-C side information.

Type CRTCSI and press F10, then specify values for **Side information** (14), **Remote location** (11), **Transaction program** (15), **Local location** (3), **Mode**, and **Remote network identifier** (9) and press Enter.

```
Create Comm Side Information (CRTCSI)

Type choices, press Enter.

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

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


Additional Parameters

Device . . . . . .  *LOC     Name, *LOC
Local location . . . . . .  AS400LU    Name, *LOC, *NETATR
Mode . . . . . .  #INTER    Name, *NETATR
Remote network identifier . . .  NETID     Name, *LOC, *NETATR, *NONE
Authority . . . . . .  *LIBCRTAUT  Name, *LIBCRTAUT, *CHANGE...

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

d) Add a communications entry for APPC.

At a command-line, type ADDCMNE and press Enter, then specify values for **Subsystem description** (5) and **Device** (13), and press Enter again.

```
Add Communications Entry (ADDCMNE)

Type choices, press Enter.

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



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

e) Add a configuration list entry.

Type ADDCFGLE *APPNRMT and press F4, then specify values for **Remote location name** (11), **Remote network identifier** (9), **Local location name** (3), **Remote control point** (10), and **Control point net ID** (9) and press Enter.

```
Add Configuration List Entries (ADDCFGLE)

Type choices, press Enter.

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

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

What to do next

The LU 6.2 connection is now established. You are ready to complete the configuration as described in "Configuring the channels on IBM i" on page 26.

Related tasks

"Establishing a TCP connection on IBM i" on page 24
If TCP is already configured, there are no extra configuration tasks. If TCP/IP is not configured you need to add a TCP/IP interface, add a TCP/IP loopback interface, and add a default route.

IBM i *Configuration parameters for an LU 6.2 connection on IBM i*

The parameters needed to set up communication from IBM MQ on an IBM i system to one of the other IBM MQ platforms using an LU 6.2 connection.

Use these tables with the tables for the platform to which you are connecting.

Where numbers appear in the *Reference* column, they indicate that the value must match that in the appropriate table elsewhere in this section. The task steps in "Establishing an LU 6.2 connection on IBM i" on page 16 refer to the values in the *ID* column of this table.

The entries in the *Parameter name* column are explained in "Explanation of terms used in the tables" on page 23.

Definition for the local node

Table 7. Configuration examples for the definition for the local node

ID	Parameter name	Reference	Example used
1	Local network ID		**NETID**
2	Local control point name		**AS400PU**
3	LU name		**AS400LU**
4	LAN destination address		**10005A5962EF**
5	Subsystem description		**QCMN**
6	Line description		**TOKENRINGL**
7	Resource name		**LIN041**

Table 7. Configuration examples for the definition for the local node (continued)			
ID	**Parameter name**	**Reference**	**Example used**
8	Local Transaction Program name		**MQSERIES**

Connection to IBM MQ on Windows

Windows

Windows

Table 8. Configuration examples for connecting to IBM MQ on Windows			
ID	**Parameter name**	**Reference**	**Example used**
9	Network ID	2	**NETID**
10	Control point name	3	**WINNTCP**
11	LU name	5	**WINNTLU**
12	Controller description		**WINNTCP**
13	Device		**WINNTLU**
14	Side information		**NTCPIC**
15	Transaction Program	7	**MQSERIES**
16	LAN adapter address	9	**08005AA5FAB9**
17	Mode	17	**#INTER**

Connection to IBM MQ on AIX

AIX

Table 9. Configuration examples for connecting to IBM MQ on AIX system			
ID	**Parameter name**	**Reference**	**Example used**
9	Network ID	1	**NETID**
10	Control point name	2	**AIXPU**
11	LU name	4	**AIXLU**
12	Controller description		**AIXPU**
13	Device		**AIXLU**
14	Side information		**AIXCPIC**
15	Transaction Program	6	**MQSERIES**
16	LAN adapter address	8	**123456789012**
17	Mode	14	**#INTER**

Connection to IBM MQ on Linux (x86 platform)

Linux

Table 10. Configuration examples for connecting to IBM MQ on Linux (x86 platform)			
ID	**Parameter name**	**Reference**	**Example used**
9	Network ID	4	**NETID**
10	Control point name	2	**LINUXPU**
11	LU name	5	**LINUXLU**
12	Controller description		**LINUXPU**
13	Device		**LINUXLU**
14	Side information		**LXCPIC**
15	Transaction Program	7	**MQSERIES**
16	LAN adapter address	8	**08005AC6DF33**
17	Mode	6	**#INTER**

Connection to IBM MQ for z/OS

Table 11. Configuration examples for connection to IBM MQ for z/OS			
ID	**Parameter name**	**Reference**	**Example used**
9	Network ID	2	**NETID**
10	Control point name	3	**MVSPU**
11	LU name	4	**MVSLU**
12	Controller description		**MVSPU**
13	Device		**MVSLU**
14	Side information		**MVSCPIC**
15	Transaction Program	7	**MQSERIES**
16	LAN adapter address	8	**400074511092**
17	Mode	6	**#INTER**

Connection to a VSE/ESA system

Table 12. Configuration examples for connection to a VSE/ESA system			
ID	**Parameter Name**	**Reference**	**Example Used**
9	Network ID	1	**NETID**
10	Control point name	2	**VSEPU**
11	LU name	3	**VSELU**
12	Controller description		**VSEPU**
13	Device		**VSELU**
14	Side information		**VSECPIC**
15	Transaction Program	4	**MQ01**
16	LAN adapter address	5	**400074511092**

Table 12. Configuration examples for connection to a VSE/ESA system (continued)			
ID	**Parameter Name**	**Reference**	**Example Used**
17	Mode		**#INTER**

Explanation of terms used in the tables

1 2 3
For information about how to find the configured values, see "How to find network attributes" on page 23.

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

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

6 Line description
If this parameter has been specified it is indicated in the Description field of the resource Resource name. For more information, see "How to find the value of Resource name" on page 24. If the value is not specified, you need to create a line description.

7 Resource name
For information about how to find the configured value, see "How to find the value of Resource name" on page 24.

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

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

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

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

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

How to find network attributes

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

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

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




More...
Press Enter to continue.

F3=Exit  F12=Cancel
```

Check that the values for **Local network ID** (1), **Local control point name** (2), and **Default local location** (3), correspond to the values in the table, or your own values if you have changed them.

How to find the value of Resource name

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

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

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

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









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

IBM i Establishing a TCP connection on IBM i

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

Procedure

1. Add a TCP/IP interface.

At a command-line, type ADDTCPIFC and press Enter, then specify the **IP address** and **Line description**, and a **Subnet mask** of the machine and press Enter again.

```
Add TCP/IP Interface (ADDTCPIFC)

Type choices, press Enter.

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



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

2. Add a TCP/IP loopback interface.

   At the command-line, type ADDTCPIFC and press Enter, then specify the values for **IP address**, **Line description**, and **Subnet mask**.

```
Add TCP Interface (ADDTCPIFC)

Type choices, press Enter.

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



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

3. Add a default route.

   At the command line, type ADDTCPRTE and press Enter, then specify the values appropriate to your network and press Enter to create a default route entry.

```
Add TCP Route (ADDTCPRTE)

Type choices, press Enter.

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



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

What to do next

The TCP connection is now established. You are ready to complete the configuration as described in "Configuring the channels on IBM i" on page 26.

Related tasks

"Establishing an LU 6.2 connection on IBM i" on page 16
To establish an LU 6.2 connection on IBM i, you need to configure the local node and connect it to a partner node.

IBM i Configuring the channels on IBM i

To configure IBM MQ for the example configuration on IBM i, complete the basic configuration steps for the queue manager, then configure the sender and receiver channels.

About this task

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

Start the TCP channel listener using the command **STRMQMLSR**.

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

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

Procedure

1. Create a queue manager.
   a) Type CRTMQM and press Enter.

```
                    Create Message Queue Manager (CRTMQM)

Type choices, press Enter.

Message Queue Manager name . . .

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

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

Default transmission queue . . .   *NONE

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




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

b) In the **Message Queue Manager name** field, type AS400 and in the **Undelivered message queue** field, type DEAD.LETTER.QUEUE then press Enter.

c) Start the queue manager by entering STRMQM MQMNAME(AS400).

d) Create the undelivered message queue using the following parameters:

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

For more information and an example of how to define a queue, see step “2” on page 27.

2. Define a queue.

a) Type CRTMQMQ on the command line.

```
Create MQM Queue (CRTMQMQ)

Type choices, press Enter.

Queue name . . . . . .

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









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

b) Complete the two fields of this panel and press Enter.

Another panel is then shown, with entry fields for the other parameters you have. Defaults can be taken for all other queue attributes.

3. Define a channel.
   a) Type CRTMQMCHL on the command line to display the **Create MQM Channel** panel.

```
Create MQM Channel (CRTMQMCHL)

Type choices, press Enter.

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









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

   b) Complete the two fields of this panel and press Enter.

   Another panel is then displayed on which you can specify the values for the other parameters given earlier. Defaults can be taken for all other channel attributes.

4. Configure the channels for the example configuration.

   For more information about the parameters used in the following examples, see "Channel configuration parameters for IBM i" on page 29.

   Windows These examples are for connecting IBM MQ on IBM i with IBM MQ on Windows. To connect to IBM MQ on another platform, use the appropriate values from the tables in "Channel configuration parameters for IBM i" on page 29 instead of the values for Windows.

   a) Define the sender channel as shown in the following examples:

   - Using SNA:

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

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

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

   - Using TCP:

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

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

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

b) Define the receiver channel as shown in the following examples:

- Using SNA:

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

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

- Using TCP:

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

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

IBM i *Channel configuration parameters for IBM i*

The parameters needed to configure the channels for the example configuration on IBM i.

Step “4” on page 28 of “Configuring the channels on IBM i” on page 26 describes the configuration to be performed on the IBM i queue manager to implement the channel described in “Example: setting up cross-platform communication for IBM MQ” on page 5. The examples in “Configuring the channels on IBM i” on page 26 are for connecting IBM MQ for IBM i and IBM MQ for Windows. To connect to IBM MQ on another platform, use the values from the appropriate table in place of the values for Windows.

Note:

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

For more information and examples of how to create the objects listed in the tables, see “Configuring the channels on IBM i” on page 26.

Definition for local node

Table 13. Configuration examples for the definition for the local node			
ID	**Parameter name**	**Reference**	**Example used**
A	Queue Manager Name		**AS400**
B	Local queue name		**AS400.LOCALQ**

Connection to IBM MQ on Windows

Windows

The values in this section of the table must match the values used in "Channel configuration parameters for Windows" on page 42, as indicated.

Table 14. Configuration examples for connecting to IBM MQ on Windows			
ID	**Parameter name**	**Reference**	**Example used**
C	Remote queue manager name	A	**WINNT**
D	Remote queue name		**WINNT.REMOTEQ**
E	Queue name at remote system	B	**WINNT.LOCALQ**
F	Transmission queue name		**WINNT**
G	Sender (SNA) channel name		**AS400.WINNT.SNA**
H	Sender (TCP/IP) channel name		**AS400.WINNT.TCP**
I	Receiver (SNA) channel name	G	**WINNT.AS400.SNA**
J	Receiver (TCP/IP) channel name	H	**WINNT.AS400.TCP**

Connection to IBM MQ on AIX

AIX

The values in this section of the table must match the values used in "Channel configuration parameters for AIX" on page 12, as indicated.

Table 15. Configuration examples for connecting to IBM MQ on AIX			
ID	**Parameter name**	**Reference**	**Example used**
C	Remote queue manager name	A	**AIX**
D	Remote queue name		**AIX.REMOTEQ**
E	Queue name at remote system	B	**AIX.LOCALQ**
F	Transmission queue name		**AIX**
G	Sender (SNA) channel name		**AS400.AIX.SNA**
H	Sender (TCP/IP) channel name		**AS400.AIX.TCP**
I	Receiver (SNA) channel name	G	**AIX.AS400.SNA**
J	Receiver (TCP) channel name	H	**AIX.AS400.TCP**

Connection to IBM MQ on Linux

Linux

The values in this section of the table must match the values used in “Channel configuration parameters for Linux” on page 36, as indicated.

Table 16. Configuration examples for connecting to IBM MQ on Linux

ID	Parameter name	Reference	Example used
C	Remote queue manager name	A	**LINUX**
D	Remote queue name		**LINUX.REMOTEQ**
E	Queue name at remote system	B	**LINUX.LOCALQ**
F	Transmission queue name		**LINUX**
G	Sender (SNA) channel name		**AS400.LINUX.SNA**
H	Sender (TCP/IP) channel name		**AS400.LINUX.TCP**
I	Receiver (SNA) channel name	G	**LINUX.AS400.SNA**
J	Receiver (TCP/IP) channel name	H	**LINUX.AS400.TCP**

Connection to IBM MQ for z/OS

The values in this section of the table must match the values used in “Channel configuration parameters for z/OS” on page 47, as indicated.

Table 17. Configuration examples for connecting to IBM MQ for z/OS

ID	Parameter name	Reference	Example used
C	Remote queue manager name	A	**MVS**
D	Remote queue name		**MVS.REMOTEQ**
E	Queue name at remote system	B	**MVS.LOCALQ**
F	Transmission queue name		**MVS**
G	Sender (SNA) channel name		**AS400.MVS.SNA**
H	Sender (TCP) channel name		**AS400.MVS.TCP**
I	Receiver (SNA) channel name	G	**MVS.AS400.SNA**
J	Receiver (TCP) channel name	H	**MVS.AS400.TCP**

Connection to a VSE/ESA system

The values in this section of the table must match the values used in your VSE/ESA system.

Table 18. Configuration examples for connecting to a VSE/ESA system

ID	Parameter name	Reference	Example used
C	Remote queue manager name	A	**VSE**
D	Remote queue name		**VSE.REMOTEQ**
E	Queue name at remote system	B	**VSE.LOCALQ**
F	Transmission queue name		**VSE**
G	Sender channel name		**AS400.VSE.SNA**

Table 18. Configuration examples for connecting to a VSE/ESA system (continued)			
ID	**Parameter name**	**Reference**	**Example used**
I	Receiver channel name	G	**VSE.AS400.SNA**

Linux Example: setting up IBM MQ cross-platform communication on Linux

This example shows how to set up communication links from IBM MQ on Linux to IBM MQ on another platform and establish a working channel to that platform.

Before you begin

For background information about this example and how to use it, see "Example: setting up cross-platform communication for IBM MQ" on page 5 and "How to use the cross-platform communication examples" on page 7.

About this task

This example covers setting up cross platform communication from IBM MQ on Linux to the following platforms:

- Windows Windows
- AIX AIX
- IBM i IBM i
- z/OS z/OS

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

Establish a network connection using either LU 6.2 or TCP.

Note: For TCP, 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.

Procedure

1. Establish a network connection using LU6.2

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

   For the latest information about configuring SNA over TCP/IP, refer to the the Administration Guide for your version of Linux from the following documentation: Communications Server for Data Center Deployment on Linux library.
2. Establish a TCP connection using the inet daemon (INETD)
   a) 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.
   b) Edit the file `/etc/inetd.conf`.
   If you do not have the following line in that file, add it as shown:

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

c) Find the process ID of the inetd with the command:

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

d) Run the command:

```
kill -1 inetd processid
```

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

For example:

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

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

This avoids error messages being generated if there is a limitation on the number of outstanding connection requests queued at a single TCP port. For information about the number of outstanding connection requests, see Using the TCP listener backlog option.

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

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

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

3. Establish a TCP connection 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.

a) 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
```

If you do not have the following line in the file, add it as shown:

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

b) Create a file called IBM MQ in the XINETD configuration directory, `/etc/xinetd.d` by adding the following stanza to the file:

```
# IBM MQ service for XINETD
service MQSeries
{
  disable         = no
  flags           = REUSE
```

```
  socket_type     = stream
  wait            = no
  user            = mqm
  server          = MQ_INSTALLATION_PATH/bin/amqcrsta
  server_args     = -m queue.manager.name
  log_on_failure += USERID
}
```

c) Restart the extended inet daemon by issuing the following command:

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

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

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

```
cps = 500 60
```

4. Complete the configuration now that the TCP/IP connection is established.
   Go to “Configuring the channels on Linux” on page 34.

Linux Configuring the channels on Linux

To configure IBM MQ for the example configuration on Linux, complete the basic configuration steps for the queue manager, then configure the sender and receiver channels.

Before you begin

Before beginning the 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
```

About this task

Notes:

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

Procedure

1. Setup the basic configuration:
   a) Create the queue manager and a set of default objects, 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

b) 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.

2. Configure the channels for the example configuration.

For more information about the parameters used in the following examples, see "Channel configuration parameters for Linux" on page 36. In each case, the example shows the MQSC command. Either start **runmqsc** from an Linux command line and enter each command in turn, or build the commands into a command file.

Windows These examples are for connecting IBM MQ on Linux with IBM MQ on Windows. To connect to IBM MQ on another platform, use the appropriate values from the tables in "Channel configuration parameters for Linux" on page 36 instead of the values for Windows.

a) Define the sender channel as shown in the following examples:

- Using SNA

```
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 (LINUX.WINNT.SNA) chltype(sdr) +        G
    trptype(lu62) +
    conname('WINNTCPIC') +                      14
    xmitq(WINNT) +                              F
    replace
```

- Using TCP

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

b) Defining the receiver channel as shown in the following examples:

- Using SNA:

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

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

- Using TCP:

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

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

Linux *Channel configuration parameters for Linux*

The parameters needed to configure the channels for the example configuration on Linux.

Step "2" on page 35 of "Configuring the channels on Linux" on page 34 describes the configuration to be performed on the Linux queue manager to implement the channel described in "Example: setting up cross-platform communication for IBM MQ" on page 5. The examples in "Configuring the channels on Linux" on page 34 are for connecting IBM MQ for IBM i and IBM MQ for Windows. To connect to IBM MQ on another platform, use the values from the appropriate table in place of the values for Windows.

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

Definition for local node

Table 19. Configuration examples for the definition for the local node

ID	Parameter name	Reference	Example used
A	Queue Manager Name		**LINUX**
B	Local queue name		**LINUX.LOCALQ**

Connection to IBM MQ on Windows

Windows

The values in this section of the table must match those used in "Channel configuration parameters for Windows" on page 42, as indicated.

Table 20. Configuration examples for connecting to IBM MQ on Windows

ID	Parameter name	Reference	Example used
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 IBM MQ on AIX

AIX

The values in this section of the table must match those used in "Configuring the channels on AIX" on page 10, as indicated.

Table 21. Configuration examples for connecting to IBM MQ on AIX

ID	Parameter name	Reference	Example used
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/IP) channel name		**LINUX.AIX.TCP**
I	Receiver (SNA) channel name	G	**AIX.LINUX.SNA**
J	Receiver (TCP/IP) channel name	H	**AIX.LINUX.TCP**

Connection to IBM MQ for IBM i

IBM i

The values in this section of the table must match those used in "Channel configuration parameters for IBM i" on page 29, as indicated.

Table 22. Configuration examples for connecting to IBM MQ on IBM i

ID	Parameter Name	Reference	Example Used
C	Remote queue manager name	A	**AS400**
D	Remote queue name		**AS400.REMOTEQ**
E	Queue name at remote system	B	**AS400.LOCALQ**
F	Transmission queue name		**AS400**
G	Sender (SNA) channel name		**LINUX.AS400.SNA**
H	Sender (TCP) channel name		**LINUX.AS400.TCP**
I	Receiver (SNA) channel name	G	**AS400.LINUX.SNA**
J	Receiver (TCP) channel name	H	**AS400.LINUX.TCP**

Connection to IBM MQ for z/OS

z/OS

The values in this section of the table must match those used in "Channel configuration parameters for z/OS" on page 47, as indicated.

Table 23. Configuration examples for connecting to IBM MQ for z/OS			
ID	**Parameter name**	**Reference**	**Example used**
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**
J	Receiver (TCP) channel name	H	**MVS.LINUX.TCP**

Connection to IBM MQ for z/OS using queue sharing groups

z/OS

The values in this section of the table must match those used in “Shared channel configuration parameters” on page 56, as indicated.

Table 24. Configuration examples for connecting to IBM MQ for z/OS using queue sharing groups			
ID	**Parameter name**	**Reference**	**Example used**
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		**LINUX.QSG.SNA**
H	Sender (TCP) channel name		**LINUX.QSG.TCP**
I	Receiver (SNA) channel name	G	**QSG.LINUX.SNA**
J	Receiver (TCP) channel name	H	**QSG.LINUX.TCP**

Windows Example: setting up IBM MQ cross-platform communication on Windows

This example shows how to set up communication links from IBM MQ on Windows to IBM MQ on another platform and establish a working channel to that platform.

Before you begin

For background information about this example and how to use it, see “Example: setting up cross-platform communication for IBM MQ” on page 5 and “How to use the cross-platform communication examples” on page 7.

About this task

This example covers setting up cross platform communication from IBM MQ on Windows to the following platforms:

- AIX
- IBM i
- Linux
- z/OS

Procedure

1. Establish a network connection using LU6.2.

   See AnyNet® SNA over TCP/IP and Communications Server for Windows for information about configuring AnyNet SNA over TCP/IP.
2. Establish a network connection using TCP.

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

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

   Use the following command to start the IBM MQ TCP listener is:

   ```
   runmqlsr -t tcp
   ```
3. Establish a network connection using NetBIOS.

   a) At each end of the channel, specify the local NetBIOS name to be used by the IBM MQ channel processes in the queue manager configuration file qm.ini.

   For example, the NETBIOS stanza in Windows at the sending end might look like the following:

   ```
   NETBIOS:
    LocalName=WNTNETB1
   ```

   and at the receiving end, look like the following:

   ```
   NETBIOS:
    LocalName=WNTNETB2
   ```

   Each IBM MQ process must use a different local NetBIOS name. Do not use your system name as the NetBIOS name because Windows already uses it.

   b) At each end of the channel, verify the LAN adapter number being used on your system.

   The IBM MQ for Windows default for logical adapter number 0 is NetBIOS running over an Internet Protocol network. To use native NetBIOS you must select logical adapter number 1. See Establishing the LAN adapter number.

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

   ```
   NETBIOS:
    AdapterNum=1
   ```

   c) So that sender channel initiation works, specify the local NetBIOS name by the MQNAME environment variable:

   ```
   SET MQNAME=WNTNETB1I
   ```

   Note: This name must be unique.

   d) 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.

e) At the receiving end, define the corresponding receiver channel.

For example:

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

f) Start the channel initiator.

Each new channel is started as a thread rather than as a new process:

```
runmqchi
```

g) At the receiving end, start the IBM MQ listener:

```
runmqlsr -t netbios
```

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

4. Complete the configuration now that the network connection is established. See “Configuring the channels on Windows” on page 40.

Windows Configuring the channels on Windows

To configure IBM MQ for the example configuration on Windows, complete the basic configuration steps for the queue manager, then configure the sender and receiver channels.

About this task

Notes:

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

```
AMQSBCG q_name qmgr_name
```

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

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

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

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

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

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

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

Procedure

1. To set up the basic configuration by using the command prompt, complete the following steps:
   a) Create the queue manager and a set of default objects 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

   b) Start the queue manager using the command:

```
strmqm winnt
```

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

2. Configure the channels for the example configuration.

   For more information about the parameters used in the following examples, see "Channel configuration parameters for Windows" on page 42. In each case, the example shows the MQSC command. Either start **runmqsc** from an Linux command line and enter each command in turn, or build the commands into a command file. These examples are for connecting IBM MQ for Windows and IBM MQ for AIX. To connect to IBM MQ on another platform, use the appropriate values from the tables in "Channel configuration parameters for Windows" on page 42 instead of the values for IBM MQ for AIX.

   a) Define the sender channel as shown in the following examples:

   - Using SNA

```
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
```

   - Using TCP

```
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
```

b) Define the receiver channel as shown in the following examples:

- Using SNA:

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

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

- Using TCP:

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

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

What to do next

Automatic startup

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

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

For more information, see Administering IBM MQ.

Running channels as processes or threads

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

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

Windows *Channel configuration parameters for Windows*

The parameters needed to configure the channels for the example configuration on Windows.

Step "2" on page 41 of "Configuring the channels on Windows" on page 40 describes the configuration to be performed on the Linux queue manager to implement the channel described in "Example: setting up cross-platform communication for IBM MQ" on page 5. The examples in "Configuring the channels on Windows" on page 40 are for connecting IBM MQ for Windows and IBM MQ for AIX. To connect to IBM MQ on another platform, use the values from the appropriate table in place of the values for Windows.

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

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

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

Definition for local node

Table 25. Configuration examples for the definition for the local node

ID	Parameter name	Reference	Example used
A	Queue Manager Name		**WINNT**
B	Local queue name		**WINNT.LOCALQ**

Connection to IBM MQ on AIX

AIX

The values in this section of the table must match those used in "Channel configuration parameters for AIX" on page 12, as indicated.

Table 26. Configuration examples for connecting to IBM MQ on AIX

	Parameter Name	Reference	Example Used
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 IBM MQ on IBM i

IBM i

The values in this section of the table must match those used in "Channel configuration parameters for IBM i" on page 29, as indicated.

Table 27. Configuration examples for connecting to IBM MQ on IBM i

ID	Parameter Name	Reference	Example Used
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**

Table 27. Configuration examples for connecting to IBM MQ on IBM i (continued)			
ID	**Parameter Name**	**Reference**	**Example Used**
H	Sender (TCP) channel name		**WINNT.AS400.TCP**
I	Receiver (SNA) channel name	G	**AS400.WINNT.SNA**
J	Receiver (TCP) channel name	H	**AS400.WINNT.TCP**

Connection to IBM MQ for z/OS

z/OS

The values in this section of the table must match those used in "Channel configuration parameters for z/OS" on page 47, as indicated.

Table 28. Configuration examples for connecting to IBM MQ for z/OS			
ID	**Parameter name**	**Reference**	**Example used**
C	Remote queue manager name	A	**MVS**
D	Remote queue name		**MVS.REMOTEQ**
E	Queue name at remote system	B	**MVS.LOCALQ**
F	Transmission queue name		**MVS**
G	Sender (SNA) channel name		**WINNT.MVS.SNA**
H	Sender (TCP) channel name		**WINNT.MVS.TCP**
I	Receiver (SNA) channel name	G	**MVS.WINNT.SNA**
J	Receiver (TCP) channel name	H	**MVS.WINNT.TCP**

Connection to IBM MQ for z/OS using queue sharing groups

z/OS

The values in this section of the table must match those used in "Shared channel configuration parameters" on page 56, as indicated.

Table 29. Configuration examples for connecting to IBM MQ for z/OS using queue sharing groups			
ID	**Parameter name**	**Reference**	**Example used**
C	Remote queue manager name	A	**QSG**
D	Remote queue name		**QSG.REMOTEQ**
E	Queue name at remote system	B	**QSG.SHAREDQ**
F	Transmission queue name		**QSG**
G	Sender (SNA) channel name		**WINNT.QSG.SNA**
H	Sender (TCP) channel name		**WINNT.QSG.TCP**
I	Receiver (SNA) channel name	G	**QSG.WINNT.SNA**
J	Receiver (TCP) channel name	H	**QSG.WINNT.TCP**

z/OS Example: setting up IBM MQ cross-platform communication on z/OS

This example shows how to set up communication links from IBM MQ on z/OS to IBM MQ on another platform and establish a working channel to that platform.

Before you begin

For background information about this example and how to use it, see “Example: setting up cross-platform communication for IBM MQ” on page 5 and “How to use the cross-platform communication examples” on page 7.

About this task

This example covers setting up cross platform communication from IBM MQ on z/OS to the following platforms:

- Windows Windows
- AIX AIX
- Linux Linux
- IBM i IBM i
- VSE/ESA

You can also connect any of the following:

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

Procedure

1. Establish a network connection.
   - Establishing an LU 6.2 connection

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

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

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

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

   See “Configuring the channels on IBM MQ for z/OS ” on page 46 for details on how you configure the channels.

z/OS Configuring the channels on IBM MQ for z/OS

To configure IBM MQ for the example configuration on z/OS, start and configure the channels and listeners.

Procedure

1. Start the channel initiator using the command:

```
/cpf   START CHINIT  1
```

2. Start an LU 6.2 listener using the command:

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

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

3. Start a TCP listener using the command:

```
/cpf   START LSTR
```

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

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

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

4. Configure the channels for the example configuration.

   For more information about the parameters used in the following examples, see “Channel configuration parameters for z/OS” on page 47. These examples are for connecting IBM MQ for z/OS and IBM MQ for Windows. To connect to IBM MQ on another platform use the values from the appropriate table in “Channel configuration parameters for z/OS” on page 47 instead of the values for Windows.

   a) Define the sender channel as shown in the following example:s

   For LU 6.2:

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

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

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

   For TCP:

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

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

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

b) Define the receiver channel as shown in the following examples:

For LU 6.2:

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

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

For TCP:

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

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

z/OS *Channel configuration parameters for z/OS*

The parameters needed to configure the channels for the example configuration on z/OS.

Step “4” on page 46 of “Configuring the channels on IBM MQ for z/OS ” on page 46 describes the configuration to be performed on the z/OS queue manager to implement the channel described in “Example: setting up cross-platform communication for IBM MQ” on page 5. The examples in “Configuring the channels on IBM MQ for z/OS ” on page 46 are for connecting IBM MQ for z/OS and IBM MQ for Windows. To connect to IBM MQ on another platform use the values from the appropriate table in place of the values for Windows.

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

Definition for local node

Table 30. Configuration examples for the definition for the local node

ID	Parameter Name	Reference	Example Used
A	Queue Manager Name		**MVS**
B	Local queue name		**MVS.LOCALQ**

Connection to IBM MQ on Windows

Windows

The values in this section of the table must match the values used in “Channel configuration parameters for Windows” on page 42, as indicated.

Table 31. Configuration examples for connecting to IBM MQ on Windows			
ID	**Parameter Name**	**Reference**	**Example Used**
C	Remote queue manager name	A	**WINNT**
D	Remote queue name		**WINNT.REMOTEQ**
E	Queue name at remote system	B	**WINNT.LOCALQ**
F	Transmission queue name		**WINNT**
G	Sender (LU 6.2) channel name		**MVS.WINNT.SNA**
H	Sender (TCP) channel name		**MVS.WINNT.TCP**
I	Receiver (LU 6.2) channel name	G	**WINNT.MVS.SNA**
J	Receiver (TCP/IP) channel name	H	**WINNT.MVS.TCP**

Connection to IBM MQ on AIX

AIX

The values in this section of the table must match the values used in "Channel configuration parameters for AIX" on page 12, as indicated.

Table 32. Configuration examples for connecting to IBM MQ on AIX			
ID	**Parameter Name**	**Reference**	**Example Used**
AIX AIX ***Connection to IBM MQ for AIX***			
C	Remote queue manager name	A	**AIX**
D	Remote queue name		**AIX.REMOTEQ**
E	Queue name at remote system	B	**AIX.LOCALQ**
F	Transmission queue name		**AIX**
G	Sender (LU 6.2) channel name		**MVS.AIX.SNA**
H	Sender (TCP/IP) channel name		**MVS.AIX.TCP**
I	Receiver (LU 6.2) channel name	G	**AIX.MVS.SNA**
J	Receiver (TCP/IP) channel name	H	**AIX.MVS.TCP**

Connection to IBM MQ on Linux

Linux

The values in this section of the table must match the values used in "Channel configuration parameters for Linux" on page 36, as indicated.

Table 33. Configuration examples for connecting to IBM MQ on Linux			
ID	**Parameter Name**	**Reference**	**Example Used**
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**

Table 33. Configuration examples for connecting to IBM MQ on Linux (continued)			
ID	**Parameter Name**	**Reference**	**Example Used**
G	Sender (LU 6.2) channel name		**MVS.LINUX.SNA**
H	Sender (TCP) channel name		**MVS.LINUX.TCP**
I	Receiver (LU 6.2) channel name	G	**LINUX.MVS.SNA**
J	Receiver (TCP/IP) channel name	H	**LINUX.MVS.TCP**

Connection to IBM MQ on IBM i

IBM i

The values in this section of the table must match the values used in "Channel configuration parameters for IBM i" on page 29, as indicated.

Table 34. Configuration examples for connecting to IBM MQ on IBM i			
ID	**Parameter name**	**Reference**	**Example used**
C	Remote queue manager name	A	**AS400**
D	Remote queue name		**AS400.REMOTEQ**
E	Queue name at remote system	B	**AS400.LOCALQ**
F	Transmission queue name		**AS400**
G	Sender (LU 6.2) channel name		**MVS.AS400.SNA**
H	Sender (TCP/IP) channel name		**MVS.AS400.TCP**
I	Receiver (LU 6.2) channel name	G	**AS400.MVS.SNA**
J	Receiver (TCP/IP) channel name	H	**AS400.MVS.TCP**

z/OS Example: setting up IBM MQ cross-platform communication on z/OS using QSGs

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

Before you begin

Setting up communication links from a queue sharing group to a platform other than z/OS is the same as described in "Example: setting up IBM MQ cross-platform communication on z/OS" on page 45.

For background information about this example and how to use it, see "Example: setting up cross-platform communication for IBM MQ" on page 5 and "How to use the cross-platform communication examples" on page 7.

Procedure

1. Establish a network connection using one of the following options.
   - Establish an LU 6.2 connection as described in "Establishing an LU 6.2 connection into a queue sharing group" on page 50.
   - Establish a TCP connection using Sysplex Distributor as described in "Establishing a TCP connection using Sysplex Distributor" on page 54.
2. Define some channels to complete the configuration after the connection is established.

See “Configuring shared channels on IBM MQ for z/OS ” on page 54 for details of this process.

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

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

About this task

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

Procedure

1. Use VTAM Generic Resources to have one connection name to connect to the queue sharing group.
   a) SYS1.PARMLIB(APPCPMxx) contains the start-up parameters for APPC. You must add a line to this file to tell APPC where to locate the sideinfo.

      This line must be of the form:

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

   b) Add another line to SYS1.PARMLIB(APPCPMxx) to define the local LU name you intend to use for the IBM MQ LU 6.2 group listener.

      The line you add must take the form

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

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

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

   c) Start the APPC subsystem with the command:

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

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

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

      ```
      SET APPC=xx
      ```

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

   d) Add the new LU to a suitable VTAM major node definition. These are typically in SYS1.VTAMLST.

      The APPL definition will look like the sample shown.

      ```
      MVSLU APPL  ACBNAME=MVSLU1,     9
                  APPXC=YES,
                  AUTOSES=0,
                  DDRAINL=NALLOW,
                  DLOGMOD=#INTER,     6
                  DMINWML=10,
                  DMINWNR=10,
                  DRESPL=NALLOW,
      ```

```
                    DSESLIM=60,
                    LMDENT=19,
                    MODETAB=MTCICS,
                    PARSESS=YES,
                    VERIFY=NONE,
                    SECACPT=ALREADYV,
                    SRBEXIT=YES
```

e) Activate the major node.

You can do this activation with the command:

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

f) Add entries defining your LU and generic resource name to the CPI-C side information data set, using the APPC utility program ATBSDFMU to do so.

Sample JCL is in *thlqual*.SCSQPROC(CSQ4SIDE) (where *thlqual* is the target library high-level qualifier for IBM MQ data sets in your installation.)

The entries you add will look like this example:

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

g) Alter the queue manager object to use the correct distributed queuing parameters using the following command.

You must specify the local LU (9) assigned to your queue manager in the LUGROUP attribute of the queue manager.

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

2. Define a connection to a partner by adding an entry to the CPI-C side information data set.

a) Add an entry to the CPI-C side information data set to define the connection.

Sample JCL to do this definition is in *thlqual*.SCSQPROC(CSQ4SIDE).

The entry you add looks like this:

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

What to do next

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

Go to “Configuring shared channels on IBM MQ for z/OS ” on page 54.

z/OS *Configuration parameters for an LU 6.2 connection*

The following table lists all the parameters required to set up communication from a z/OS system to IBM MQ on another platform.

The steps required to set up an LU 6.2 connection are described in “Establishing an LU 6.2 connection into a queue sharing group” on page 50, with numbered cross-references to the parameters in the example.

Numbers in the Reference column indicate that the value must match that in the appropriate example elsewhere in this section. The examples that follow in this section refer to the values in the ID column. The entries in the Parameter Name column are explained in "Explanation of terms" on page 53.

Definition for local node using generic resources

Table 35. Configuration examples for the definition for the local node using generic resources

ID	Parameter name	Reference	Example used
1	Command prefix		**/cpf**
2	Network ID		**NETID**
3	Node name		**MVSPU**
6	Modename		**#INTER**
7	Local Transaction Program name		**MQSERIES**
8	LAN destination address		**400074511092**
9	Local LU name		**MVSLU1**
10	Generic resource name		**MVSGR**
11	Symbolic destination		**G1**
12	Symbolic destination for generic resource name		**G2**

Connection to IBM MQ on Windows

Windows

Table 36. Configuration examples for connecting to IBM MQ on Windows using LU 6.2

ID	Parameter name	Reference	Example used
13	Symbolic destination		**M3**
14	Modename	21	**#INTER**
15	Remote Transaction Program name	7	**MQSERIES**
16	Partner LU name	5	**WINNTLU**
21	Remote node ID	4	**05D 30F65**

Connection to IBM MQ on AIX

AIX

Table 37. Configuration examples for connecting to IBM MQ on AIX using LU 6.2

ID	Parameter name	Reference	Example used
13	Symbolic Destination		**M4**
14	Modename	18	**#INTER**
15	Remote Transaction Program name	6	**MQSERIES**
16	Partner LU name	4	**AIXLU**

Explanation of terms

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

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

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

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

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

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

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

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

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

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

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

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

z/OS Establishing a TCP connection using Sysplex Distributor

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

Procedure

1. Define a Distributed DVIPA address as follows:
   a) Add a DYNAMICXCF statement to the IPCONFIG. This statement is used for inter-image connectivity using dynamically created XCF TCP/IP links.
   b) Use the VIPADYNAMIC block on each image in the Sysplex.

      On the owning image, code a VIPADEFINE statement to create the DVIPA Then code a VIPADISTRIBUTE statement to distribute it to all other or selected images.

      On the backup image, code a VIPABACKUP statement for the DVIPA address.
2. Add the SHAREPORT option for the port to be shared in the PORT reservation list in the PROFILE data set if more than one channel initiator is to be started on any LPAR in the sysplex.

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

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

What to do next

Go to "Configuring shared channels on IBM MQ for z/OS " on page 54.

z/OS Configuring shared channels on IBM MQ for z/OS

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

About this task

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

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

Procedure

1. Start the channel initiator using the command:

```
/cpf   START CHINIT
```

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

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

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

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

4. Configure the channels for the example configuration.

For more information about the parameters used in the following examples, see “Shared channel configuration parameters” on page 56. These examples are for connecting IBM MQ for z/OS and Windows. To connect to IBM MQ on another platform, use the appropriate values from the tables in “Shared channel configuration parameters” on page 56 instead of the values for Windows.

a) Define the shared sender channel as shown in the following examples.

Using LU 6.2:

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

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

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

Using TCP

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

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

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

b) Define the shared receiver channel as shown in the following examples.

Using LU 6.2:

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

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

Using TCP:

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

```
            Disposition :   SHARED

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

z/OS *Shared channel configuration parameters*

The parameters needed to configure a shared channel for the example configuration on z/OS.

Step “4” on page 54 of “Configuring shared channels on IBM MQ for z/OS ” on page 54 describes the configuration to be performed on the z/OS queue manager to implement the channel described in “Example: setting up cross-platform communication for IBM MQ” on page 5. The examples in “Configuring shared channels on IBM MQ for z/OS ” on page 54 are for connecting IBM MQ for z/OS and Windows. To connect to IBM MQ on another platform, use the values from the appropriate table in place of the values for Windows.

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

Definition for local node

Table 38. Configuration examples for the definition for the local node

ID	Parameter Name	Reference	Example Used
A	Queue Manager Name		**QSG**
B	Local queue name		**QSG.SHAREDQ**

Connection to IBM MQ on Windows

Windows

The values in this section of the table must match the values used in “Channel configuration parameters for Windows” on page 42, as indicated.

Table 39. Configuration examples for connecting to to IBM MQ on Windows

ID	Parameter name	Reference	Example used
C	Remote queue manager name	A	**WINNT**
D	Remote queue name		**WINNT.REMOTEQ**
E	Queue name at remote system	B	**WINNT.LOCALQ**
F	Transmission queue name		**WINNT**
G	Sender (LU 6.2) channel name		**QSG.WINNT.SNA**
H	Sender (TCP) channel name		**QSG.WINNT.TCP**
I	Receiver (LU 6.2) channel name	G	**WINNT.QSG.SNA**
J	Receiver (TCP/IP) channel name	H	**WINNT.QSG.TCP**

Connection to IBM MQ on AIX

AIX

The values in this section of the table must match the values used in “Channel configuration parameters for AIX” on page 12, as indicated.

Table 40. Configuration examples for connecting to IBM MQ on AIX			
ID	**Parameter name**	**Reference**	**Example used**
C	Remote queue manager name		**AIX**
D	Remote queue name		**AIX.REMOTEQ**
E	Queue name at remote system	B	**AIX.LOCALQ**
F	Transmission queue name		**AIX**
G	Sender (LU 6.2) channel name		**QSG.AIX.SNA**
H	Sender (TCP/IP) channel name		**QSG.AIX.TCP**
I	Receiver (LU 6.2) channel name	G	**AIX.QSG.SNA**
J	Receiver (TCP/IP) channel name	H	**AIX.QSG.TCP**

z/OS Example: setting up IBM MQ cross-platform communication for intra-group queuing on z/OS

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

About this task

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

Procedure

1. Set up and run Configuration 1.
   For more information, see “Setting up and running configuration 1” on page 58.
2. Set up and run Configuration 2.
   For more information, see “Setting up and running configuration 2” on page 60.
3. Set up and run Configuration 3.
   For more information, see “Setting up and running configuration 3” on page 62.

What to do next

You can expand the example in a number of ways by:

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

z/OS Setting up and running configuration 1

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

About this task

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

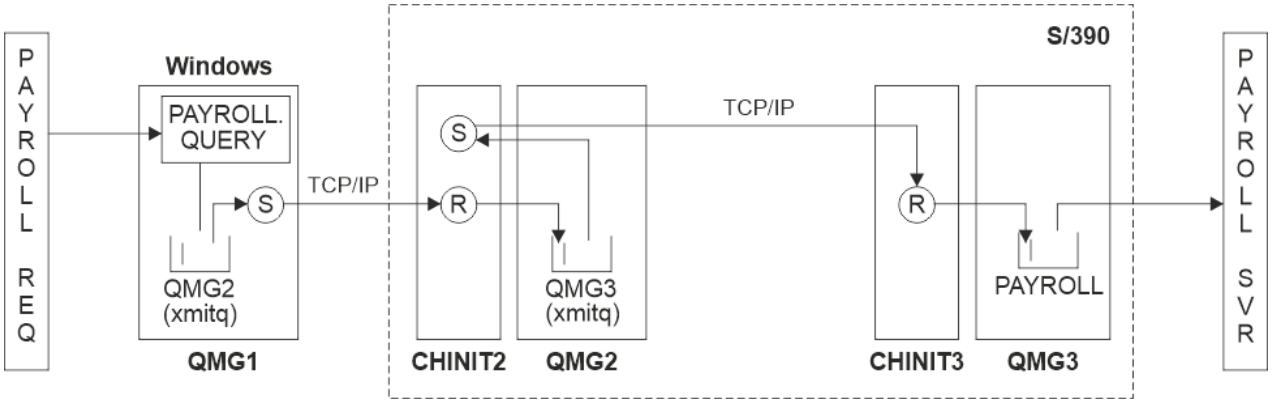

Figure 2. Configuration 1: z/OS using intra-group queuing

The flow of operations is as follows:

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

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

Procedure

1. Procedure on QMG1:
   a) Setup the remote queue definition:

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

   b) Setup the transmission queue definition:

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

c) Setup the sender channel definition using TCP/IP:

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

Note: Replace MVSQMG2(1415) with your queue manager connection name and port.

d) Setup the receiver channel definition using TCP/IP:

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

e) Setup the reply-to queue definition:

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

2. Procedure on QMG2:

a) Setup the transmission queue definition:

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

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

b) Setup the sender channel definitions using TCP/IP:

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

Note: Replace WINTQMG1(1414) with your queue manager connection name and port.

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

Note: Replace MVSQMG3(1416) with your queue manager connection name and port.

c) Setup the receiver channel definitions using TCP/IP:

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

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

3. Procedure on QMG3:

a) Setup the local queue definition:

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

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

b) Setup the sender channel definition using TCP/IP:

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

Note: Replace MVSQMG2(1415) with your queue manager connection name and port.

c) Setup the receiver channel definition using TCP/IP:

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

4. Use the following procedure to run configuration 1:
   a) Start queue managers QMG1, QMG2, and QMG3.
   b) Start channel initiators for QMG2 and QMG3.
   c) Start the listeners on QMG1 to listen to port 1414, QMG2 to listen on port 1415, and QMG3 to listen on port 1416.
   d) Start sender channels on QMG1, QMG2, and QMG3.
   e) Start the payroll query requesting application connected to QMG1.
   f) Start the payroll server application connected to QMG3.
   g) Submit a payroll query request to QMG3 and wait for the payroll reply.

z/OS Setting up and running configuration 2

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

About this task

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

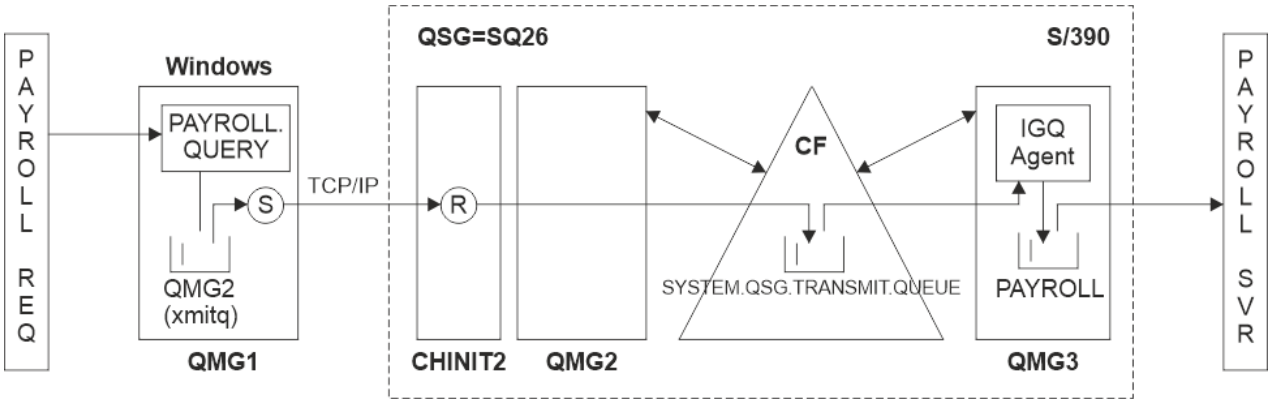

Figure 3. Configuration 2

The flow of operations is as follows:

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

4. Receiver channel (R) on queue manager QMG2 puts the query on to queue PAYROLL on queue manager QMG3. As queue PAYROLL on QMG3 resolves to shared transmission queue SYSTEM.QSG.TRANSMIT.QUEUE, the query is put on to shared transmission queue SYSTEM.QSG.TRANSMIT.QUEUE.
5. IGQ agent on queue manager QMG3 retrieves the query from shared transmission queue SYSTEM.QSG.TRANSMIT.QUEUE, and puts it on to local queue PAYROLL on queue manager QMG3.
6. The payroll server application connected to queue manager QMG3 retrieves the query from local queue PAYROLL, processes it, and generates a suitable reply.

Notes:

- The payroll query example transfers small messages only. If you need to transfer both persistent and non-persistent messages, you can establish a combination of Configuration 1 and Configuration 2, so that large messages can be transferred using the distributed queuing route, while small messages can be transferred using the potentially faster intra-group queuing route.
- The definitions do not take into account triggering, and that only channel definitions for communication using TCP/IP are provided.
- The example assumes that you have already configured queue managers QMG2 and QMG3 to be members of the same queue sharing group.

Procedure

1. Procedure on QMG1:
   a) Setup the remote queue definition:

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

   b) Setup the transmission queue definition:

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

   c) Setup the sender channel definition for TCP/IP:

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

   Note: Replace MVSQMG2(1415) with your queue manager connection name and port.

   d) Setup the receiver channel definition for TCP/IP:

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

   e) Setup the reply-to queue definition:

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

2. Procedure on QMG2:
   a) Setup the transmission queue definition:

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

DEFINE QLOCAL(SYSTEM.QSG.TRANSMIT.QUEUE) QSGDISP(SHARED) +
DESCR('IGQ Transmission queue') REPLACE PUT(ENABLED) USAGE(XMITQ) +
```

```
GET(ENABLED) INDXTYPE(CORRELID) CFSTRUCT('APPLICATION1') +
DEFSOPT(SHARED) DEFPSIST(NO)
```

Note: Replace APPLICATION1 with your defined CF structure name. Also, this queue being a shared queue, need only be defined on one of the queue managers in the queue sharing group.

b) Setup the sender channel definitions for TCP/IP:

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

Note: Replace WINTQMG1(1414) with your queue manager connection name and port.

c) Setup the receiver channel definition for TCP/IP:

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

d) Setup the queue manager definition:

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

3. Procedure on QMG3:

a) Setup the local queue definition:

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

b) Setup the queue manager definition:

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

4. Use the following procedure to run configuration 2:

a) Start queue managers QMG1, QMG2, and QMG3.

b) Start the channel initiator for QMG2.

c) Start the listeners on QMG1 to listen on port 1414, and QMG2 to listen on port 1415.

d) Start the sender channel on QMG1 and QMG2.

e) Start the payroll query requesting application connected to QMG1.

f) Start the payroll server application connected to QMG3.

g) Submit a payroll query request to QMG3 and wait for the payroll reply.

z/OS Setting up and running configuration 3

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

About this task

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

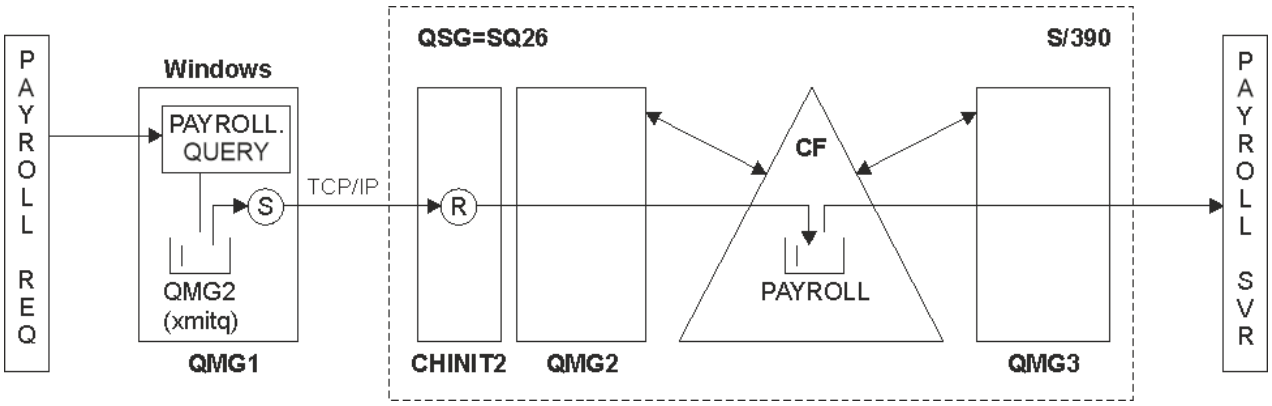

Figure 4. Configuration 3

The flow of operations is:

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

This configuration is certainly the simplest to configure. However, you would need to configure distributed queuing or intra-group queuing to transfer replies (generated by the payroll server application connected to queue manager QMG3) from queue manager QMG3 to queue manager QMG2, and then on to queue manager QMG1.

For the configuration used to transfer replies back to the payroll request application, see "Example: planning a message channel for z/OS using queue sharing groups" on page 159.

Notes:

- Only channel definitions for communication using TCP/IP are provided.
- The example assumes that you have already configured queue managers QMG2 and QMG3 to be members of the same queue sharing group.
- No definitions are required on QMG3.

Procedure

1. Procedure on QMG1:
   a) Setup the remote queue definition:

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

   b) Setup the transmission queue definition:

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

   c) Setup the sender channel definition:

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

Note: Replace MVSQMG2(1415) with your queue manager connection name and port.

d) Setup the transmission channel definition:

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

e) Setup the reply-to queue definition:

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

2. Procedure on QMG2:

a) Setup the transmission queue definition:

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

b) Setup the sender channel definitions for TCP/IP:

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

Note: Replace WINTQMG1(1414) with your queue manager connection name and port.

c) Setup the receiver channel definitions for TCP/IP:

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

d) Setup the local queue definition:

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

Note: Replace APPLICATION1 with your defined CF structure name. Also this queue being a shared queue, need only be defined on one of the queue managers in the queue sharing group.

3. Use the following procedure to run configuration 3:

a) Start queue managers QMG1, QMG2, and QMG3.

b) Start the channel initiator for QMG2.

c) Start the listeners on QMG1 to listen on port 1414, and QMG2 to listen on port 1415.

d) Start sender channels on QMG1 and QMG2.

e) Start the payroll query requesting application connected to QMG1.

f) Start the payroll server application connected to QMG3.

g) Submit a payroll query request to QMG3 and wait for the payroll reply.

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

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

crtmqdir command

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

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

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

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

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

Access is granted in the following three categories:

mqm group only

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

The file permissions for these files and directories are:

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

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

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

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

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

The file permissions for these files and directories are:

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

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

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

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

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

IBM MQ 8.0:

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

All users read and write access

Files that have read and write access for all users

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

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

These special files are:

Symbolic links

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

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

Socket files

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

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

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

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

IBM MQ 8.0:

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

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

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

Directories that have read and write access for all users

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

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

/var/mqm/errors/

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

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

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

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

AMQERR0*.LOG

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

AMQnnnnn.*.FDC

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

/var/mqm/trace/

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

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

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

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

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

AMQnnnnn.*.TRC

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

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

IBM MQ 8.0:

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

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

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

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

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

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

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

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

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

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

HOME

A `${HOME}/.mqm` directory is created when using an unregistered or non-installed version of IBM MQ, such as the redistributable client.

The directory is created so that IBM MQ has a reliable way of accessing its socket files using a path that fits within the **sun_path** length. If IBM MQ cannot write to the HOME directory you receive an error message.

Use of System V IPC resources by IBM MQ

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

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

- Check the ownership.

  The owning user of IBM MQ System V IPC resources is always the 'mqm' user on AIX and Linux platforms. On IBM i the owning user is 'QMQM'.

- IBM MQ 8.0 and later, use the `amqspdbg` utility.

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

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

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

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

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

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

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

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

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

Cause of the problem

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

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

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

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

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

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

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

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

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

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

Resolving the problem

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

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

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

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

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

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

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

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

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

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

   For more information see “IBM MQ file system permissions applied to /var/mqm” on page 65.

Related concepts

Filesystem

Windows IBM MQ file system permissions on Windows

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

Data directory

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

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

Administrators
Full control

mqm group
Full control

SYSTEM
Full control

Everyone
Read and execute

The exceptions are:

DATADIR\errors
Everyone full control

DATADIR\trace
Everyone full control

DATADIR\log

Administrators
Full control

mqm group
Full control

SYSTEM
Full control

Everyone
Read

DATADIR\log\<qmgrname>\active

Administrators
Full control

mqm group
Full control

SYSTEM
Full control

No access granted to Everyone.

The error log files AMQERR01.LOG, and so on, do not inherit their security settings from their directory but are instead set to `Everyone: Full Control`.

Earlier releases of the product

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

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

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

Naming restrictions for queues

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

Restrictions on name lengths

Queues can have names up to 48 characters long.

Reserved queue names

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

Table 41. Reserved queue names and descriptions

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

Table 41. Reserved queue names and descriptions (continued)

Queue Name	Description
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 MQ channel authentication data queue
SYSTEM.CICS.INITIATION.QUEUE	Queue used for triggering (not for z/OS)
SYSTEM.CLUSTER.COMMAND.QUEUE	Queue used to communicate repository changes between queue managers
SYSTEM.CLUSTER.HISTORY.QUEUE	The queue is used to store the history of cluster state information for service purposes.
SYSTEM.CLUSTER.REPOSITORY.QUEUE	Queue used to hold information about the repository
SYSTEM.CLUSTER.TRANSMIT.MODEL.QUEUE	The queue used to create individual transmit queues for each cluster-sender channel.
SYSTEM.CLUSTER.TRANSMIT.QUEUE	Transmission queue for all destinations managed by cluster support
SYSTEM.COMMAND.INPUT	Queue to which command messages are sent on z/OS
SYSTEM.COMMAND.REPLY.MODEL	Model queue definition for command replies (for z/OS)
SYSTEM.DEAD.LETTER.QUEUE	Dead-letter queue (not for z/OS)
SYSTEM.DEFAULT.ALIAS.QUEUE	Default alias queue definition
SYSTEM.DEFAULT.INITIATION.QUEUE	Queue used to trigger a specified process (not for z/OS)
SYSTEM.DEFAULT.LOCAL.QUEUE	Default local queue definition
SYSTEM.DEFAULT.MODEL.QUEUE	Default model queue definition
SYSTEM.DEFAULT.REMOTE.QUEUE	Default remote queue definition
SYSTEM.DURABLE.SUBSCRIBER.QUEUE	A local queue used to hold a persistent copy of the durable subscriptions in the queue manager
SYSTEM.HIERARCHY.STATE	Queue used to hold information about the state of inter-queue manager relationships in a publish/subscribe hierarchy
SYSTEM.JMS.TEMPQ.MODEL	Model for JMS temporary queues
SYSTEM.INTERNAL.REPLY.QUEUE	IBM MQ internal reply queue (not for z/OS)

Table 41. Reserved queue names and descriptions (continued)	
Queue Name	**Description**
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 MQ Explorer
SYSTEM.MQSC.REPLY.QUEUE	Model queue definition for MQSC command replies (not for z/OS)
SYSTEM.QSG.CHANNEL.SYNCQ	Shared local queue used for storing messages that contain the synchronization information for shared channels (z/OS only)
SYSTEM.QSG.TRANSMIT.QUEUE	Shared local queue used by the intra-group queuing agent when transmitting messages between queue managers in the same queue sharing group (z/OS only)
SYSTEM.RETAINED.PUB.QUEUE	A local queue used to hold a copy of each retained publication in the queue manager.
SYSTEM.SELECTION.EVALUATION.QUEUE	IBM MQ internal selection evaluation queue (not for z/OS)
SYSTEM.SELECTION.VALIDATION.QUEUE	IBM MQ internal selection validation queue (not for z/OS)

Naming restrictions for other objects

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

Restrictions on name length

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

Channels can have names up to 20 characters long.

Storage classes can have names up to 8 characters long.

CF structures can have names up to 12 characters long.

Reserved object names

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

Table 42. Reserved object names and descriptions	
Object Name	**Description**
SYSTEM.ADMIN.SVRCONN	Server-connection channel used for remote administration of a queue manager

Table 42. Reserved object names and descriptions (continued)	
Object Name	**Description**
SYSTEM.AUTO.RECEIVER	Default receiver channel for auto definition (AIX, Linux, and Windows systems only)
SYSTEM.AUTO.SVRCONN	Default server-connection channel for auto definition (Multiplatforms 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
SYSTEM.DEFAULT.AUTHINFO.OCSP	Default authentication information object definition for defining authentication information objects of type OCSP
SYSTEM.DEFAULT.LISTENER.LU62	Default SNA listener (Windows only)
SYSTEM.DEFAULT.LISTENER.NETBIOS	Default NetBIOS listener (Windows only)
SYSTEM.DEFAULT.LISTENER.SPX	Default SPX listener (Windows only)
SYSTEM.DEFAULT.LISTENER.TCP	Default TCP/IP listener (Multiplatforms only)
SYSTEM.DEFAULT.NAMELIST	Default namelist definition
SYSTEM.DEFAULT.PROCESS	Default process definition
SYSTEM.DEFAULT.SEVICE	Default service (Multiplatforms only)
SYSTEM.DEFAULT.TOPIC	Default topic definition
SYSTEM.QPUBSUB.QUEUE.NAMELIST	A list of queues for the Queued Publish/Subscribe interface to monitor
z/OS z/OS SYSTEMST	Default storage class definition (z/OS only)

Queue name resolution

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 distributed queue management, which ensures that queue name resolution is performed by queue managers at both sending and receiving ends of a channel.

The main benefits of this approach are as follows:

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

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

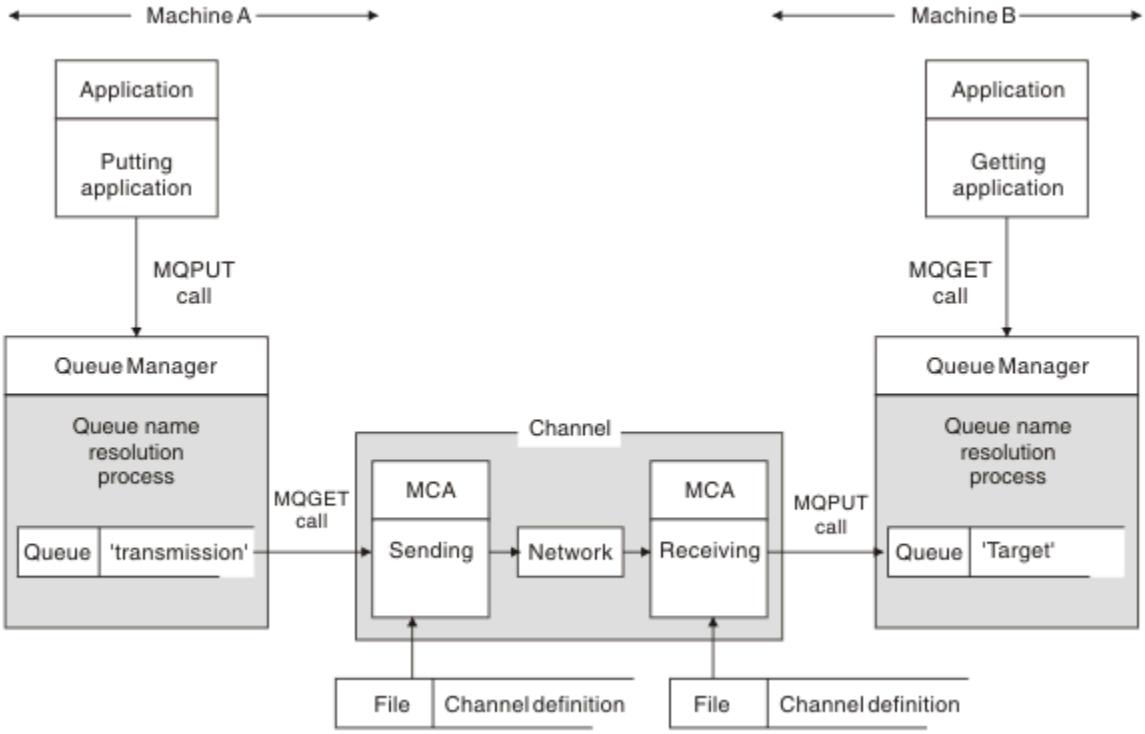

Figure 5. Name resolution

Referring to Figure 5 on page 75, 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.

Related tasks

Putting messages on remote queues

What is queue name resolution?

Queue name resolution is vital to distributed queue management. It removes the need for applications to be concerned with the physical location of queues, and insulates applications from 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.

To uncouple from the application design the exact path over which the data travels, there is 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.

How are destination object attributes resolved for aliases, remote queues and cluster queues?

When name resolution is performed on behalf of an application API call, attributes affecting the use of the object are resolved from a combination of the originally named object, the "path" (see “Queue name resolution” on page 74), and the resolved target object. In a queue manager cluster, the "named object" in question is the clustered object (queue or topic) definition. This is a subset of the object attributes shared between queue managers and visible through. for example, **DISPLAY QCLUSTER**.

Where an attribute can be defined on the named object opened by the application, this takes precedence. For example, all DEF**** attributes (default persistence, priority, and asynchronous put response) can be configured on alias and remote queue definitions. These take effect when the alias or remote queue is opened by an application, rather than any resolved destination queue or transmission queue.

Attributes designed to restrict or limit application interaction with a target object cannot typically be defined on the named object (remote queue definition or alias). For example, **MAXMSGL** and **MAXDEPTH** cannot be set on a remote queue definition or alias, and are not passed between members of a queue manager cluster. These attributes are therefore taken from the resolved queue (for example the local queue, appropriate transmission queue, or SYSTEM.CLUSTER.TRANSMIT.QUEUE). On arrival at a remote queue manager, a second constraint might be applied on delivery to the target queue, which could result in a message being placed on a dead letter queue, or the channel being forced to stop.

Note that a special case of attribute resolution is **PUT** and **GET** enablement. For both of these attributes, any instance of **DISABLED** in the queue path results in an overall resolved attribute of **DISABLED**.

System and default objects

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

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

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

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

Note: There are two other default objects not included in the tables: the queue manager object and the object catalogue. These are objects in the sense that they are logged and recoverable.

- System and default objects: queues
- System and default objects: topics
- System and default objects: server channels
- System and default objects: client channels
- System and default objects: authentication information
- System and default objects: communications information
- System and default objects: listeners
- System and default objects: namelists
- System and default objects: processes
- System and default objects: services

Table 43. System and default objects: queues

Object name	Description
SYSTEM.ADMIN.ACCOUNTING.QUEUE	Queue for accounting message data generated when an application disconnects from the queue manager.

Table 43. System and default objects: queues (continued)

Object name	Description
SYSTEM.ADMIN.ACTIVITY.QUEUE	Queue that holds returned activity report messages.
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.LOGGER.EVENT	Event queue for logger event (journal receiver) messages.
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 MQI, queue and channel 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.AMQP.COMMAND.QUEUE	IBM MQ Administration Command Queue for AMQP
SYSTEM.AUTH.DATA.QUEUE	The queue that holds access control lists for the queue manager. Used by the object authority manager (OAM).
SYSTEM.BROKER.ADMIN.STREAM	Administration stream for queued Pub/Sub interface
SYSTEM.BROKER.CONTROL.QUEUE	Publish/subscribe interface control queue.
SYSTEM.BROKER.DEFAULT.STREAM	Default stream for queued Pub/Sub interface
SYSTEM.BROKER.INTER.BROKER.COMMUNICATIONS	Broker to broker communications queue.
SYSTEM.CHANNEL.INITQ	Channel initiation queue.
SYSTEM.CHANNEL.SYNCQ	The queue that holds the synchronization data for channels.
SYSTEM.CHLAUTH.DATA.QUEUE	IBM MQ channel authentication data queue
SYSTEM.CICS.INITIATION.QUEUE	The 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 used to store the history of cluster state information for service purposes.
SYSTEM.CLUSTER.REPOSITORY.QUEUE	The queue used to store all repository information.
SYSTEM.CLUSTER.TRANSMIT.MODEL.QUEUE	The queue used to create individual transmit queues for each cluster-sender channel.
SYSTEM.CLUSTER.TRANSMIT.QUEUE	The transmission queue for all messages to all clusters.

Table 43. System and default objects: queues (continued)	
Object name	**Description**
SYSTEM.DEAD.LETTER.QUEUE	Dead-letter (undelivered-message) queue.
SYSTEM.DEFAULT.ALIAS.QUEUE	Default alias queue.
SYSTEM.DEFAULT.INITIATION.QUEUE	Default initiation queue.
SYSTEM.DEFAULT.LOCAL.QUEUE	Default local queue.
SYSTEM.DEFAULT.MODEL.QUEUE	Default model queue.
SYSTEM.DEFAULT.REMOTE.QUEUE	Default remote queue.
SYSTEM.DOTNET.XARECOVERY.QUEUE	IBM MQ .NET XA Recovery Queue
SYSTEM.DURABLE.MODEL.QUEUE	The queue used as a model for managed durable subscriptions.
SYSTEM.DURABLE.SUBSCRIBER.QUEUE	The queue used to hold a persistent copy of the durable subscriptions in the queue manager.
SYSTEM.HIERARCHY.STATE	IBM MQ distributed publish/subscribe hierarchy relationship state.
SYSTEM.INTER.QMGR.CONTROL	IBM MQ distributed publish/subscribe control queue.
SYSTEM.INTER.QMGR.FANREQ	IBM MQ distributed publish/subscribe internal proxy subscription fan-out process input queue.
SYSTEM.INTER.QMGR.PUBS	IBM MQ distributed publish/subscribe publications.
SYSTEM.INTERNAL.REPLY.QUEUE	
SYSTEM.INTERNAL.REQUEST.QUEUE	
SYSTEM.JMS.TEMPQ.MODEL	Model for JMS temporary queues
SYSTEM.MQEXPLORER.REPLY.MODEL	The IBM MQ Explorer reply-to queue. This is a model queue that creates a temporary dynamic queue for replies to the IBM MQ Explorer.
SYSTEM.MQSC.REPLY.QUEUE	MQSC command reply-to queue. This is a model queue that creates a temporary dynamic queue for replies to remote MQSC commands.
SYSTEM.NDURABLE.MODEL.QUEUE	A queue used as a model for managed non durable subscriptions.
SYSTEM.PENDING.DATA.QUEUE	Support deferred messages in JMS.
SYSTEM.PROTECTION.ERROR.QUEUE	IBM MQ Message Protection Error Queue.
SYSTEM.PROTECTION.POLICY.QUEUE	IBM MQ Message Protection Policy Queue.
SYSTEM.REST.REPLY.QUEUE	
SYSTEM.RETAINED.PUB.QUEUE	A queue used to hold a copy of each retained publication in the queue manager.
SYSTEM.SELECTION.EVALUATION.QUEUE	
SYSTEM.SELECTION.VALIDATION.QUEUE	

Table 44. System and default objects: topics	
Object name	**Description**
SYSTEM.ADMIN.TOPIC	Administration topic.
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.BROKER.ADMIN.STREAM	Admin stream used by the queued publish/subscribe interface.
SYSTEM.BROKER.DEFAULT.STREAM	The default stream used by the queued publish/subscribe interface.
SYSTEM.BROKER.DEFAULT.SUBPOINT	The default subpoint used by the queued publish/subscribe interface.
SYSTEM.DEFAULT.TOPIC	Default topic definition.

Table 45. System and default objects: server channels	
Object name	**Description**
SYSTEM.AUTO.RECEIVER	Dynamic receiver channel.
SYSTEM.AUTO.SVRCONN	Dynamic server-connection channel.
SYSTEM.DEF.AMQP	Default AMQP channel. Note that the object is defined, but the AMQP service is not supported.
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.DEFAULT.AUTHINFO.IDPWLDAP	
SYSTEM.DEFAULT.AUTHINFO.IDPWOS	

Table 46. System and default objects: client channels	
Object name	**Description**
SYSTEM.DEF.CLNTCONN	Default client-connection channel.

Table 47. System and default objects: authentication information	
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 48. System and default objects: communications information	
Object name	**Description**
SYSTEM.DEFAULT.COMMINFO.MULTICAST	Default communications information object for multicast.

Table 49. System and default objects: listeners	
Object name	**Description**
SYSTEM.DEFAULT.LISTENER.TCP	Default listener for TCP transport.
Windows SYSTEM.DEFAULT.LISTENER.LU62	Default LU62 listener.
Windows SYSTEM.DEFAULT.LISTENER.NETBIOS	Default NETBIOS listener.
Windows SYSTEM.DEFAULT.LISTENER.SPX	Default SPX listener.

Table 50. System and default objects: namelists	
Object name	**Description**
SYSTEM.DEFAULT.NAMELIST	Default namelist definition.
SYSTEM.QPUBSUB.QUEUE.NAMELIST	A list of queue names monitored by the queued publish/subscribe interface.
SYSTEM.QPUBSUB.SUBPOINT.NAMELIST	A list of topic objects used by the queued publish/subscribe interface to match topic objects to subscription points.

Table 51. System and default objects: processes	
Object name	**Description**
SYSTEM.DEFAULT.PROCESS	Default process definition.

Table 52. System and default objects: services	
Object name	**Description**
SYSTEM.AMQP.SERVICE	MQ Light API service. Note that the object is defined, but the service is not supported.
SYSTEM.DEFAULT.SERVICE	Default service.

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 53. Default values of SYSTEM.BASE.TOPIC

Parameter	Value
TOPICSTR	"
V 9.3.1 Multi CAPEXPRY	NOLIMIT
CLROUTE	DIRECT
CLUSTER	The default value is an empty string.
COMMINFO	SYSTEM.DEFAULT.COMMINFO.MULTICAST
DEFPRESP	SYNC
DEFPRTY	0
DEFPSIST	NO
DESCR	'Base topic for resolving attributes'
DURSUB	YES
MCAST	DISABLED
MDURMDL	SYSTEM.DURABLE.MODEL.QUEUE
MNDURMDL	SYSTEM.NDURABLE.MODEL.QUEUE
NPMSGDLV	ALLAVAIL
PMSGDLV	ALLDUR
PROXYSUB	FIRSTUSE
PUB	ENABLED
PUBSCOPE	ALL
z/OS z/OS QSGDISP (z/OS platform only)	QMGR
SUB	ENABLED
SUBSCOPE	ALL
USEDLQ	YES
WILDCARD	PASSTHRU

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

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

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

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

Configuration files 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:

- mqs.ini file stanzas and attributes helps you configure the:
  - *AllQueueManagers* stanza
  - *DefaultQueueManager* stanza
  - *ExitProperties* stanza
  - *LogDefaults* stanza
  - *Security* stanza in the `qm.ini` file
- qm.ini file stanzas and attributes helps you configure the:
  - Windows *AccessMode* stanza (Windows only)
  - *Service* stanza - for Installable services
  - *Log* stanza
  - Linux AIX *RestrictedMode* stanza (AIX 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 AIX, 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
- IBM MQ MQI client configuration file, `mqclient.ini` helps you configure the:
  - *CHANNELS* stanza
  - *ClientExitPath* stanza
  - Windows *LU62*, *NETBIOS* and *SPX* stanza (Windows only)
  - *MessageBuffer* stanza

  - *SSL* stanza
  - *TCP* stanza
  - V 9.3.3 *Trace* stanza (used for IBM MQ .NET and XMS .NET only)
- "Configuration file stanzas for distributed queuing" on page 85 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.
- Using automatic configuration helps you configure the:
  - AutoConfig stanza
  - AutoCluster stanza
  - Variables stanza

Configuration files

See:

- **mqs.ini** file
- **qm.ini** file
- **mqclient.ini** file

for a list of the possible stanzas in each configuration file. Linux AIX

mqs.ini file

Example of an IBM MQ configuration file for AIX and Linux systems shows an example `mqs.ini` file.

An `mqs.ini` file can contain the following stanzas:

- *AllQueueManagers*
- *DefaultQueueManager*
- *ExitProperties*
- *LogDefaults*

In addition, there is one *QueueManager* stanza for each queue manager.

qm.ini file

Example queue manager configuration file for IBM MQ for AIX or Linux systems shows an example `qm.ini` file.

A `qm.ini` file can contain the following stanzas:

- *ExitPath*

- *Log*
- *QMErrorLog*
- *QueueManager*
- *Security*
- *ServiceComponent*

  Multi To configure *InstallableServices* use the *Service* and *ServiceComponent* stanzas.
- *Connection* for *DefaultBindType*

  Attention: You must create a *Connection* stanza if you need one.
- *SSL and TLS*
- *TCP, LU62, and NETBIOS*
- XAResourceManager

In addition, you can change the:

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

by using the crtmqm command.

mqclient.ini file

An `mqclient.ini` file can contain the following stanzas:

- *CHANNELS*
- *ClientExitPath*
- *LU62*, *NETBIOS*, and *SPX*
- *MessageBuffer*
- *SSL*
- *TCP*

In addition, you might need a PreConnect stanza to configure a preconnect exit.

Configuration file stanzas for distributed queuing

A description of the stanzas of the queue manager configuration file, qm.ini, related to distributed queuing.

This topic shows the stanzas in the queue manager configuration file that relate to distributed queuing. It applies to the queue manager configuration file for IBM MQ for Multiplatforms. The file is called `qm.ini` on all platforms.

The stanzas that relate to distributed queuing are:

- CHANNELS
- TCP
- LU62
- NETBIOS
- EXITPATH

Figure 6 on page 86 shows the values that you can set using these stanzas. When you are defining one of these stanzas, you do not need to start each item on a new line. You can use either a semicolon (;) or a hash character (#) to indicate a comment.

```
CHANNELS:
 MAXCHANNELS=n           ; Maximum number of channels allowed, the
                         ; default value is 100.
 MAXACTIVECHANNELS=n     ; Maximum number of channels allowed to be active at
                         ; any time, the default is the value of MaxChannels.
 MAXINITIATORS=n         ; Maximum number of initiators allowed, the default
                         ; and maximum value is 3.
 MQIBINDTYPE=type        ; Whether the binding for applications is to be
                         ; "fastpath" or "standard".
                         ; The default is "standard".
 PIPELINELENGTH=n        ; The maximum number of concurrent threads a channel will use.
                         ; The default is 1. Any value greater than 1 is treated as 2.
 ADOPTNEWMCA=chltype     ; Stops previous process if channel fails to start.
                         ; The default is "NO".
 ADOPTNEWMCATIMEOUT=n    ; Specifies the amount of time that the new
                         ; process should wait for the old process to end.
                         ; The default is 60.
 ADOPTNEWMCACHECK=       ; Specifies the type checking required.
           typecheck     ; The default is "NAME","ADDRESS", and "QM".
 CHLAUTHEARLYADOPT=Y/N   ; The order in which connection authentication and channel
authentication rules are
                         ; processed. If not present in the qm.ini file the default is "N".
From MQ9.0.4 all
                         ; queue managers are created with a defa\ult of "Y"
 PASSWORDPROTECTION=     ; From MQ8.0,set protected passwords in the MQCSP structure, rather
than using TLS.
             options     ; The options are "compatible", "always", "optional" and "warn"
                         ; The default is "compatible".
 IGNORESEQNUMBERMISMATCH ; How the queue manager handles a sequence number mismatch during
channel startup.
                  =Y/N   ; The options are "Y" and "N" with the default being "N".
 CHLAUTHIGNOREUSERCASE   ; Enables a queue manager to make username matching within CHLAUTH
rules case-insensitive.
                  =Y/N   ; The options are "Y" and "N" with the default being "N".
 CHLAUTHISSUEWARN=Y      ; If you want message AMQ9787 to be generated when you set theWARN=YES
attribute
                         ; on the SET CHLAUTH command.
TCP:                     ; TCP entries
 PORT=n                  ; Port number, the default is 1414
 KEEPALIVE=Yes           ; Switch TCP/IP KeepAlive on
LU62:
 LIBRARY2=DLLName2       ; Used if code is in two libraries
EXITPATH:[1]     Location of user exits
 EXITPATHS=              ; String of directory paths.
```

Figure 6. qm.ini stanzas for distributed queuing

Notes:

1. EXITPATH applies only to the following platforms:
   - AIX
   - Windows

Related tasks

Configuring

z/OS Configuring z/OS

Changing IBM MQ configuration information on Multiplatforms

Channel attributes

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

You choose the attributes of a channel to be optimal for a particular set of circumstances for each channel. However, when the channel is running, the actual values might have changed during startup negotiations. See Preparing channels.

Many attributes have default values, and you can use these values for most channels. However, in those circumstances where the defaults are not optimal, see this section for guidance in selecting the correct values.

Note: In IBM MQ for IBM i, most attributes can be specified as *SYSDFTCHL, which means that the value is taken from the system default channel in your system.

The channel types for IBM MQ channel attributes are listed in the following table, in the order of the MQSC command parameters.

Note: For cluster channels (the CLUSSDR and CLUSRCVR columns in the table), if an attribute can be set on both channels, set it on both and ensure that the settings are identical. If there is any discrepancy between the settings, those that you specify on the CLUSRCVR channel are likely to be used. This is explained in Cluster channels.

Table 54. Channel attributes for the channel types

Attribute field	MQSC command parameter	SDR	SVR	RCVR	RQSTR	CLNT-CONN	SVR-CONN	CLUS-SDR	CLUS-RCVR	AMQP
Connection Affinity	AFFINITY					Yes				
Alter date	ALTDATE	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
Alter time	ALTTIME	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
Multi AMQP keep alive	AMQPKA									Yes
Batch heartbeat interval	BATCHHB	Yes	Yes					Yes	Yes	
Batch interval	BATCHINT	Yes	Yes					Yes	Yes	
Batch limit	BATCHLIM	Yes	Yes					Yes	Yes	
Batch size	BATCHSZ	Yes	Yes	Yes	Yes			Yes	Yes	
Certificate label	CERTLABL	Yes	Yes	Yes	Yes	Yes	Yes	Yes "1" on page 90	Yes	Yes
Channel name	CHANNEL	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
Channel type	CHLTYPE	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
Client channel weight	CLNTWGHT					Yes				
Cluster namelist	CLUSNL							Yes	Yes	
Cluster	CLUSTER							Yes	Yes	
Cluster workload priority	CLWLPRTY							Yes	Yes	
Cluster workload rank	CLWLRANK							Yes	Yes	
Cluster workload weight	CLWLWGHT							Yes	Yes	
Header compression	COMPHDR	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	
Data compression	COMPMSG	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	

Table 54. Channel attributes for the channel types (continued)

Attribute field	MQSC command parameter	SDR	SVR	RCVR	RQSTR	CLNT-CONN	SVR-CONN	CLUS-SDR	CLUS-RCVR	AMQP
Connection name	CONNAME	Yes	Yes		Yes	Yes		Yes	Yes	
Convert message	CONVERT	Yes	Yes					Yes	Yes	
Default reconnection	DEFRECON					Yes				
Description	DESCR	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
Disconnect interval	DISCINT	Yes	Yes				Yes "2" on page 90	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	Yes
Long retry count	LONGRTY	Yes	Yes					Yes	Yes	
Long retry interval	LONGTMR	Yes	Yes					Yes	Yes	
Maximum instances	MAXINST						Yes			Yes
Maximum instances per client	MAXINSTC						Yes			
Maximum message length	MAXMSGL	Yes	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	Yes
LU 6.2 mode name	MODENAME	Yes	Yes		Yes	Yes		Yes	Yes	
Monitoring	MONCHL	Yes	Yes	Yes	Yes		Yes	Yes	Yes	
Message-retry exit user data	MRDATA			Yes	Yes				Yes	
Message-retry exit name	MREXIT			Yes	Yes				Yes	
Message retry count	MRRTY			Yes	Yes				Yes	
Message retry interval	MRTMR			Yes	Yes				Yes	

Table 54. Channel attributes for the channel types (continued)

Attribute field	MQSC command parameter	SDR	SVR	RCVR	RQSTR	CLNT-CONN	SVR-CONN	CLUS-SDR	CLUS-RCVR	AMQP
Message exit user data	MSGDATA	Yes	Yes	Yes	Yes			Yes	Yes	
Message exit name	MSGEXIT	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		
Port number	PORT									Yes
PROPCTL channel options for MQGMO	PROPCTL	Yes	Yes					Yes	Yes	
PUT authority	PUTAUT			Yes	Yes		Yes "2" on page 90		Yes	
Queue manager name	QMNAME					Yes				
z/OS Disposition "2" on page 90	QSGDISP	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	
Receive exit user data	RCVDATA	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	
Receive exit user name	RCVEXIT	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	
Security exit user data	SCYDATA	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	
Security exit name	SCYEXIT	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	
Send exit user data	SENDDATA	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	
Send exit name	SENDEXIT	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	

Table 54. Channel attributes for the channel types (continued)

Attribute field	MQSC command parameter	SDR	SVR	RCVR	RQSTR	CLNT-CONN	SVR-CONN	CLUS-SDR	CLUS-RCVR	AMQP
Short retry interval	SHORTTMR	Yes	Yes					Yes	Yes	
z/OS Security policy protection^{"2" on page 90}	SPLPROT	Yes	Yes	Yes	Yes					
SSL client authentication	SSLCAUTH		Yes	Yes	Yes		Yes		Yes	Yes
SSL cipher specification	SSLCIPH	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
SSL peer	SSLPEER	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
Channel statistics	STATCHL	Yes	Yes	Yes	Yes			Yes	Yes	
LU 6.2 transaction program name	TPNAME	Yes	Yes		Yes	Yes		Yes	Yes	
Topic root	TPROOT									Yes
Transport type	TRPTYPE	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	
Use client ID	USECLTID									Yes
Use dead letter queue	USEDLQ	Yes	Yes	Yes	Yes			Yes	Yes	
User ID	USERID	Yes	Yes		Yes	Yes		Yes		
Transmission queue name	XMITQ	Yes	Yes							

Notes:

1. None of the administrative interfaces allow this attribute to be inquired or set for CLUSSDR channels. You will receive an MQRCCF_WRONG_CHANNEL_TYPE message. However, the attribute is present in CLUSSDR channel objects (including MQCD structures) and a CHAD exit can set it programmatically if required.
2. z/OS Valid on z/OS only.

IBM MQ for some platforms might not implement all the attributes shown in this section. Exceptions and platform differences are mentioned in the individual attribute descriptions, where relevant.

The name of each attribute is shown in brackets.

The attributes are arranged in alphabetical order in groups.

Related reference

MQSC commands

ALTER CHANNEL

DEFINE CHANNEL

Channel attributes for MQSC keywords (A-B)

An alphabetical list of the channel attributes for MQSC keywords, starting with the letter *A* or *B*.

AFFINITY (Connection 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 (MQIACH_CONNECTION_AFFINITY) when multiple applicable channel definitions are available.

The possible values are:

PREFERRED
The first connection in a process reading a client channel definition table (CCDT) creates a list of applicable definitions based on the client channel weight, with any definitions having a weight of 0 first and in alphabetical order. Each connection in the process attempts to connect using the first definition in the list. If a connection is unsuccessful the next definition is used. Unsuccessful definitions with client channel weight values other than 0 are moved to the end of the list. Definitions with a client channel weight of 0 remain at the start of the list and are selected first for each connection.

Each client process with the same host name always creates the same list.

For client applications written in C, C++, or the .NET programming framework (including fully managed .NET), and for applications that use the IBM MQ classes for Java and IBM MQ classes for JMS, the list is updated if the CCDT has been modified since the list was created.

This value is the default, and has the value of 1.

NONE
The first connection in a process reading a CCDT creates a list of applicable definitions. All connections in a process select an applicable definition based on the client channel weight, with any definitions having a weight of 0 selected first in alphabetical order.

For client applications written in C, C++, or the .NET programming framework (including fully managed .NET), and for applications that use the IBM MQ classes for Java and IBM MQ classes for JMS, the list is updated if the CCDT has been modified since the list was created.

This attribute is valid for the client-connection channel type only.

ALTDATE (Alter date)

This attribute is the date on which the definition was last altered, in the form `yyyy-mm-dd` and is valid for all channel types.

ALTTIME (Alter time)

This attribute is the time at which the definition was last altered, in the form `hh.mm.ss` and is valid for all channel types.

AMQPKA (AMQP keep alive)

Multi

Use the **AMQPKA** attribute to specify a keep alive time for the AMQP client connection. If the AMQP client has not sent any frames within the keep alive interval, then the connection is closed.

The **AMQPKA** attribute determines the value of the idle-timeout attribute sent from IBM MQ to an AMQP client. The attribute is a period of time in milliseconds.

If **AMQPKA** is set to a value > 0, then IBM MQ flows half that value as the idle-timeout attribute. For example, a value of 10000 causes the queue manager to send an idle-timeout value of 5000. The client

should ensure that data is sent to IBM MQ at least every 10000 milliseconds. If data is not received by IBM MQ in that time, IBM MQ assumes that the client has lost its connection and forcibly closes the connection with an `amqp:resource-limit-exceeded` error condition.

A value of AUTO or 0 means the IBM MQ does not flow an idle-timeout attribute to the AMQP client.

An AMQP client can still flow an idle-timeout value of its own. If it does, IBM MQ flows data (or an empty AMQP frame) at least that frequently to inform the client that it is available.

BATCHHB (Batch heartbeat Interval)

his attribute allows a sending channel to verify that the receiving channel is still active just before committing a batch of messages.

The batch heartbeat interval thus allows the batch to be backed out rather than becoming in-doubt if the receiving channel is not active. By backing out the batch, the messages remain available for processing so they could, for example, be redirected to another channel.

If the sending channel has had a communication from the receiving channel within the batch heartbeat interval, the receiving channel is assumed to be still active, otherwise a 'heartbeat' is sent to the receiving channel to check. The sending channel waits for a response from the receiving end of the channel for an interval, based on the number of seconds specified in the channel Heartbeat Interval (HBINT) attribute.

The value is in milliseconds and must be in the range zero through 999999. A value of zero indicates that batch heart beating is not used.

This attribute is valid for channel types of:

- Sender
- Server
- Cluster sender
- Cluster receiver

BATCHINT (Batch interval)

This attribute is a period, in milliseconds, during which the channel keeps a batch open even if there are no messages on the transmission queue.

You can specify any number of milliseconds, from zero through 999 999 999. The default value is zero.

If you do not specify a batch interval, the batch closes when one of the following conditions is met:

- The number of messages specified in BATCHSZ have been sent.
- The number of bytes specified in BATCHLIM have been sent.
- The transmission queue is empty.

On channels with a light load, where the transmission queue frequently becomes empty, the effective batch size might be much smaller than BATCHSZ.

You can use the BATCHINT attribute to make your channels more efficient by reducing the number of short batches. Be aware, however, that you can slow down the response time, because batches last longer and messages remain uncommitted for longer.

If you specify a BATCHINT, batches close only when one of the following conditions is met:

- The number of messages specified in BATCHSZ have been sent.
- The number of bytes specified in BATCHLIM have been sent.
- There are no more messages on the transmission queue and a time interval of BATCHINT has elapsed while waiting for messages (since the first message of the batch was retrieved).

Note: BATCHINT specifies the total amount of time that is spent waiting for messages. It does not include the time spent retrieving messages that are already available on the transmission queue, or the time spent transferring messages.

This attribute is valid for channel types of:

- Sender
- Server
- Cluster sender
- Cluster receiver

BATCHLIM (Batch limit)

This attribute is the limit, in kilobytes, of the amount of data that can be sent through a channel before taking a sync point.

A sync point is taken after the message that caused the limit to be reached has flowed across the channel.

The value must be in the range 0 - 999999. The default value is 5000.

A value of zero in this attribute means that no data limit is applied to batches over this channel.

The batch is ended when one of the following conditions is met:

- BATCHSZ messages have been sent.
- BATCHLIM bytes have been sent.
- The transmission queue is empty and BATCHINT is exceeded.

This attribute is valid for channel types of:

- Sender
- Server
- Cluster sender
- Cluster receiver

BATCHSZ (Batch size)

This attribute is the maximum number of messages to be sent before a sync point is taken.

The batch size does not affect the way the channel transfers messages; messages are always transferred individually, but are committed or backed out as a batch.

To improve performance, you can set a batch size to define the maximum number of messages to be transferred between two *sync points*. The batch size to be used is negotiated when a channel starts, and the lower of the two channel definitions is taken. On some implementations, the batch size is calculated from the lowest of the two channel definitions and the two queue manager MAXUMSGS values. The actual size of a batch can be less; for example, a batch completes when there are no messages left on the transmission queue or the batch interval expires.

A large value for the batch size increases throughput, but recovery times are increased because there are more messages to back out and send again. The default BATCHSZ is 50, and you are advised to try that value first. You might choose a lower value for BATCHSZ if your communications are unreliable, making the need to recover more likely.

Sync point procedure needs a unique logical unit of work identifier to be exchanged across the link every time a sync point is taken, to coordinate batch commit procedures.

If the synchronized batch commit procedure is interrupted, an *in-doubt* situation might arise. In-doubt situations are resolved automatically when a message channel starts. If this resolution is not successful, manual intervention might be necessary, using the RESOLVE command.

Some considerations when choosing the number for batch size:

- If the number is too large, the amount of queue space taken up on both ends of the link becomes excessive. Messages take up queue space when they are not committed, and cannot be removed from queues until they are committed.

- If there is likely to be a steady flow of messages, you can improve the performance of a channel by increasing the batch size because fewer confirm flows are needed to transfer the same quantity of bytes.
- If message flow characteristics indicate that messages arrive intermittently, a batch size of 1 with a relatively large disconnect time interval might provide a better performance.
- The number can be in the range 1 through 9999.
- Even though nonpersistent messages on a fast channel do not wait for a sync point, they do contribute to the batch-size count.

This attribute is valid for channel types of:

- Sender
- Server
- Receiver
- Requester
- Cluster sender
- Cluster receiver

Channel attributes for MQSC keywords (C)

An alphabetical list of the channel attributes for MQSC keywords, starting with the letter *C*.

CERTLABL (Certificate label)

This attribute specifies the certificate label of the channel definition.

The label identifies which personal certificate in the key repository is sent to the remote peer. The certificate is defined as described in Digital certificate labels.

Inbound channels (including RCVR, RQSTR, CLUSRCVR, unqualified SERVER, and SVRCONN channels) will only send the configured certificate if the IBM MQ version of the remote peer fully supports certificate label configuration and the channel is using a TLS CipherSpec.

If that is not the case, the queue manager **CERTLABL** attribute determines the certificate sent. This restriction is because the certificate label selection mechanism for inbound channels depends upon a TLS protocol extension that is not supported in all cases. In particular, Java clients and JMS clients do not support the required protocol extension and will only ever receive the certificate configured by the queue manager **CERTLABL** attribute, regardless of the channel-specific label setting.

An unqualified server channel is one that does not have the CONNAME field set.

None of the administrative interfaces allow this attribute to be inquired or set for CLUSSDR channels. You will receive an MQRCCF_WRONG_CHANNEL_TYPE message. However, the attribute is present in CLUSSDR channel objects (including MQCD structures) and a CHAD exit can set it programmatically if required.

For more information about what the certificate label can contain, see Digital certificate labels, understanding the requirements.

This attribute is valid for all channel types.

Note: For SSL/TLS, the CERTLABL must be defined on the QMGR definition. You can, optionally, set a CERTLABL on the CHANNEL definition.

The queue manager CERTLABL is checked and must be a valid personal certificate, even if you are setting a CERTLABL on the CHANNEL definition.

CHANNEL (Channel name)

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.

CHLTYPE (Channel type)

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 (AIX, Linux, and Windows only)

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

The two ends of a channel must have the same name and have compatible types:

- Sender with receiver
- Requester with server
- Requester with sender (for callback)
- Server with receiver (server is used as a sender)
- Client-connection with server-connection
- Cluster-sender with cluster-receiver
- AMQP with AMQP

CLNTWGHT (Client channel weight)

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.

Note: When a JSON CCDT is used it is possible to have multiple channels with the same name. If multiple channels with the same name exist, and they have CLNTWGHT(0) then the channels will be selected in the order that they are defined in the JSON CCDT.

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.

CLUSNL (Cluster namelist)

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 (Cluster)

This attribute is the name of the cluster to which the channel belongs.

The maximum length is 48 characters conforming to the rules for naming IBM MQ objects.

Up to one of the resultant values of CLUSTER or CLUSNL can be non-blank. If one of the values is non-blank, the other must be blank.

This attribute is valid only for channel types of:

- Cluster sender
- Cluster receiver

CLWLPRTY (Cluster workload priority)

The CLWLPRTY channel attribute specifies the priority order for channels for cluster workload distribution. The value must be in the range 0-9, where 0 is the lowest priority and 9 is the highest.

Use the CLWLPRTY channel attribute to set a priority order for the available cluster destinations. IBM MQ selects the destinations with the highest priority before selecting destinations with the lowest cluster

destination priority. If there are multiple destinations with the same priority, it selects the least recently used destination.

If there are two possible destinations, you can use this attribute to allow failover. Messages go to the queue manager with the highest priority channel. If it becomes unavailable then messages go to the next highest priority queue manager. Lower priority queue managers act as reserves.

IBM MQ checks channel status before prioritizing the channels. Only available queue managers are candidates for selection.

Notes:

- Specify this attribute on the cluster-receiver channel at the target queue manager. Any balancing you specify on the matching cluster-sender channel is likely to be ignored. See Cluster channels.
- The availability of a remote queue manager is based on the status of the channel to that queue manager. When channels start, their state changes several times, with some of the states being less preferential to the cluster workload management algorithm. In practice this means that lower priority (backup) destinations can be chosen while the channels to higher priority (primary) destinations are starting.
- If you need to ensure that no messages go to a backup destination, do not use CLWLPRTY. Consider using separate queues, or CLWLRANK with a manual switch over from the primary to back up.

CLWLRANK (Cluster workload rank)

The **CLWLRANK** channel attribute specifies the rank of channels for cluster workload distribution. The value must be in the range 0-9, where 0 is the lowest rank and 9 is the highest.

Use the **CLWLRANK** channel attribute if you want control over the final destination for messages sent to a queue manager in another cluster. Control the choice of final destination by setting the rank of the channels connecting a queue manager to the gateway queue managers at the intersection of the clusters.

When you set **CLWLRANK**, messages take a specified route through the interconnected clusters towards a higher ranked destination. For example, messages arrive at a gateway queue manager that can send them to either of two queue managers using channels ranked 1 and 2. They are automatically sent to the queue manager connected by a channel with the highest rank, in this case the channel to the queue manager ranked 2.

IBM MQ gets the rank of channels before checking channel status. Getting the rank before checking channel status means that even non-accessible channels are available for selection. It allows messages to be routed through the network even if the final destination is unavailable.

Notes:

- Specify this attribute on the cluster-receiver channel at the target queue manager. Any balancing you specify on the matching cluster-sender channel is likely to be ignored. See Cluster channels.
- If you also used the priority attribute **CLWLPRTY**, IBM MQ selects between available destinations. If a channel is not available to the destination with the highest rank, the message is held on the transmission queue. It is released when the channel becomes available. The message does not get sent to the next available destination in the rank order.

CLWLWGHT (Cluster workload weight)

The CLWLWGHT channel attribute specifies the weight applied to CLUSSDR and CLUSRCVR channels for cluster workload distribution. The value must be in the range 1-99, where 1 is the lowest weight and 99 is the highest.

Use CLWLWGHT to send servers with more processing power more messages. The higher the channel weight, the more messages are sent over that channel.

Notes:

- Specify this attribute on the cluster-receiver channel at the target queue manager. Any balancing you specify on the matching cluster-sender channel is likely to be ignored. See Cluster channels.

- When CLWLWGHT is modified from the default of 50 on any channel, workload balancing becomes dependent on the total number of times each channel was chosen for a message sent to any clustered queue. For more information, see “The cluster workload management algorithm” on page 142.

COMPHDR (Header compression)

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.

COMPMSG (Data compression)

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 “COMPHDR (Header compression)” on page 98 for compression of the message header.

The possible values are:

NONE
No message data compression is performed. This value is the default value.

RLE
Message data compression is performed using run-length encoding.

ZLIBFAST
Message data compression is performed using the zlib compression technique. A fast compression time is preferred.

ZLIBFAST can optionally be offloaded to the zEnterprise® Data Compression facility. See zEDC Express facility for further information.

ZLIBHIGH
Message data compression is performed using the zlib compression technique. A high level of compression is preferred.

ANY
Allows the channel to support any compression technique that the queue manager supports. Only supported for Receiver, Requester and Server-Connection channels.

This attribute is valid for all channel types.

From IBM MQ 9.3.0, the ZLIBFAST and ZLIBHIGH techniques can use the hardware-accelerated zlibNX library on IBM MQ for AIX if it is installed. The zlibNX library is an enhanced version of the zlib compression library that supports hardware-accelerated data compression and decompression by using co-processors called Nest accelerators (NX) on IBM POWER9 processor-based servers. The zlibNX library is available in IBM AIX 7.2 with Technology Level 4 Expansion Pack, and later. Highly compressible messages that are over 2KB in size are most likely to benefit from opting to use the zlibNX library,

by reducing CPU usage. To enable a message channel agent (MCA) to use the zlibNX library, set the environment variable `AMQ_USE_ZLIBNX`.

CONNAME (Connection name)

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.

Multi On Multiplatforms, the TCP/IP connection name parameter of a cluster-receiver channel is optional. If you leave the connection name blank, IBM MQ generates a connection name for you, assuming the default port and using the current IP address of the system. You can override the default port number, but still use the current IP address of the system. For each connection name leave the IP name blank, and provide the port number in parentheses; for example:

```
(1415)
```

The generated **CONNAME** is always in the dotted decimal (IPv4) or hexadecimal (IPv6) form, rather than in the form of an alphanumeric DNS host name.

The maximum name length depends on the platform:

- Multi 264 characters.
- z/OS 48 characters (see note 1).

If the transport type is TCP

CONNAME is either the host name or the network address of the remote machine (or the local machine for cluster-receiver channels). For example, `(ABC.EXAMPLE.COM)`, `(2001:DB8:0:0:0:0:0:0)` or `(127.0.0.1)`. It can include the port number, for example `(MACHINE(123))`.

z/OS It can include the IP_name of a dynamic DNS group or a Network Dispatcher input port.

If you use an IPv6 address in a network that only supports IPv4, the connection name is not resolved. In a network which uses both IPv4 and IPv6, the connection name interacts with the local address to determine which IP stack is used. See “LOCLADDR (Local Address)” on page 104 for further information.

If the transport type is LU 6.2

Multi If the TPNAME and MODENAME are specified, give the fully-qualified name of the partner LU. If the TPNAME and MODENAME are blank, give the CPI-C side information object name for your specific platform.

z/OS There are two forms in which to specify the value:

- Logical unit name

  The logical unit information for the queue manager, comprising the logical unit name, TP name, and optional mode name. This name can be specified in one of three forms:

Table 55. Logical unit names and forms	
Form	**Example**
luname	IGY12355
luname/TPname	IGY12345/APING
luname/TPname/modename	IGY12345/APINGD/#INTER

For the first form, the TP name and mode name must be specified for the TPNAME and MODENAME attributes; otherwise these attributes must be blank. For client-connection channels, only the first form is allowed.

- Symbolic name

  The symbolic destination name for the logical unit information for the queue manager, as defined in the side information data set. The TPNAME and MODENAME attributes must be blank. Note that, for cluster-receiver channels, the side information is on the other queue managers in the cluster. In this case it can be a name that a channel auto-definition exit can resolve into the appropriate logical unit information for the local queue manager.

The specified or implied LU name can be that of a VTAM generic resources group.

If the transmission protocol is NetBIOS

CONNAME is the NetBIOS name defined on the remote machine.

If the transmission protocol is SPX

CONNAME is an SPX-style address consisting of a 4 byte network address, a 6 byte node address and a 2 byte socket number. Enter these values in hexadecimal, with the network and node addresses separated by a period and the socket number in brackets. For example:

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

If the socket number is omitted, the default IBM MQ SPX socket number is used. The default is X'5E86'.

This attribute is valid for channel types of:

- Sender
- Server

  It is optional for server channels, unless the server channel is triggered, in which case it must specify a connection name.
- Requester
- Client connection
- Cluster sender
- Cluster receiver

Note:

1. For name lengths, you can work around the 48 character limit in either of the following ways:
   - Set up your DNS servers so that you use, for example, host name of "myserver" instead of "myserver.location.company.com", ensuring you can use the short host name.
   - Use IP addresses.
2. The definition of transmission protocol is contained in TRPTYPE.

CONVERT (Convert message)

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

Channel attributes for MQSC keywords (D-L)

An alphabetical list of the channel attributes for MQSC keywords, starting with the letters *D* thru *L*.

DEFRECON (Default reconnection)

Specifies whether a client connection automatically reconnects a client application if its connection breaks.

The possible values are:

NO (default)
Unless overridden by **MQCONNX**, the client is not reconnected automatically.

YES
Unless overridden by **MQCONNX**, the client reconnects automatically.

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

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

This attribute is valid only for client connection channels.

DESCR (Description)

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.

DISCINT (Disconnect interval)

This attribute is the length of time after which a channel closes down, if no message arrives during that period.

This attribute is valid for channel types of:

- Sender
- Server

- Server connection using the TCP protocol only
- Cluster sender
- Cluster receiver

This attribute is a time-out, specified in seconds.

For server-to-server message channels (the server, cluster-sender, sender, and cluster-receiver channel types), 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 server-to-server message 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.

The default is 6000 seconds (100 minutes) for server-to-server message channels, and is 0 (no time-out) for server-connection channels. You can alter the default value that is used for new channels you create, by altering the default channel objects. For example, alter the DISCINT attribute on SYSTEM.DEF.SENDER to give a new default for new Sender channels you define.

For server-connection channels using the TCP protocol, the interval represents the client inactivity disconnect value, specified in seconds. If a server-connection channel program has received no communication from its partner client for this duration, it terminates the connection.

The server-connection inactivity interval applies between IBM MQ API calls from a client.

Note: A potentially long-running MQGET with wait call is not classified as inactivity and, therefore, never times out as a result of DISCINT expiring.

Attention: 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 stopping and re-starting the channel. A large value (more than an hour) might mean that system resources are consumed without benefit. 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 for server-to-server message channels is 6000 seconds (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 by altering the DISCINT attribute in the default channel definitions (for example, SYSTEM.DEF.SENDER for Sender channels) before creating your own channels.

For more information, see Stopping and quiescing channels.

HBINT (Heartbeat interval)

This attribute specifies the approximate time between heartbeat flows that are to be passed from a sending message channel agent (MCA) when there are no messages on the transmission queue.

Heartbeat flows unblock the receiving MCA, which is waiting for messages to arrive or for the disconnect interval to expire. When the receiving MCA is unblocked, it can disconnect the channel without waiting for the disconnect interval to expire. Heartbeat flows also free any storage buffers that have been allocated for large messages and close any queues that have been left open at the receiving end of the channel.

The value is in seconds and must be in the range 0 - 999 999. A value of zero means that no heartbeat flows are to be sent. The default value is 300. To be most useful, the value must be significantly less than the disconnect interval value.

With applications that use IBM MQ classes for Java, JMS or .NET APIs, the HBINT value is determined in one of the following ways:

- Either by the value on the SVRCONN channel that is used by the application.
- Or by the value on the CLNTCONN channel, if the application has been configured to use a CCDT.

For server-connection and client-connection channels, heartbeats can flow from both the server side as well as the client side independently. If no data has been transferred across the channel for the heartbeat interval, the client-connection MQI agent sends a heartbeat flow and the server-connection MQI agent responds to it with another heartbeat flow. This happens irrespective of the state of the channel, for example, irrespective of whether it is inactive while making an API call, or is inactive waiting for client user input. The server-connection MQI agent is also capable of initiating a heartbeat to the client, again irrespective of the state of the channel. To prevent both server-connection and client-connection MQI agents heart beating to each other at the same time, the server heartbeat is flowed after no data has been transferred across the channel for the heartbeat interval plus 5 seconds.

For server-connection and client-connection channels working in the channel mode before IBM WebSphere® MQ 7.0, heartbeats flow only when a server MCA is waiting for an MQGET command with the WAIT option specified, which it has issued on behalf of a client application.

For more information about making MQI channels work in the two modes, see SharingConversations (MQLONG).

KAINT (Keepalive interval)

This attribute is used to specify a timeout value for a channel.

The Keepalive Interval attribute is a value passed to the communications stack specifying the Keepalive timing for the channel. It allows you to specify a different keepalive value for each channel.

You can set the Keepalive Interval (KAINT) attribute for channels on a per-channel basis.

Multi On Multiplatforms, you can access and modify the parameter, but it is only stored and forwarded; there is no functional implementation of the parameter. If you need the functionality provided by the KAINT parameter, use the Heartbeat Interval (HBINT) parameter, as described in “HBINT (Heartbeat interval)” on page 102.

For this attribute to have any effect, TCP/IP keepalive must be enabled.

- z/OS On z/OS, you do enable keepalive by issuing the ALTER QMGR TCPKEEP(YES) MQSC command.
- Multi On Multiplatforms, it occurs when the KEEPALIVE=YES parameter is specified in the TCP stanza in the distributed queuing configuration file, `qm.ini`, or through the IBM MQ Explorer.

  Keepalive must also be enabled within TCP/IP itself, using the TCP profile configuration data set.

The value indicates a time, in seconds, and must be in the range 0 - 99999. A Keepalive Interval value of 0 indicates that channel-specific Keepalive is not enabled for the channel and only the system-wide Keepalive value set in TCP/IP is used. You can also set KAINT to a value of AUTO (this value is the default). If KAINT is set to AUTO, the Keepalive value is based on the value of the negotiated heartbeat interval (HBINT) as follows:

Table 56. Negotiated HBINT value and the corresponding KAINT value

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

This attribute is valid for all channel types.

The value is ignored for all channels that have a TransportType (TRPTYPE) other than TCP or SPX

LOCLADDR (Local Address)

This attribute specifies the local communications address for the channel.

Note: AMQP channels do not support the same format of LOCLADDR as other IBM MQ channels. For more information, see “#unique_51/unique_51_Connect_42_locladdr_amqp” on page 106.

LOCLADDR for all channels except AMQP channels

This attribute applies only 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 57. Examples of how the **LOCLADDR** parameter can be used*

LOCLADDR	**Meaning**
9.20.4.98	Channel binds to this address locally
9.20.4.98, 9.20.4.99	Channel binds to either IP address. The address might be two network adapters on one server, or a different network adapter on two different servers in a multi-instance configuration.
9.20.4.98(1000)	Channel binds to this address and port 1000 locally
9.20.4.98(1000,2000)	Channel binds to this address and uses a port in the range 1000 - 2000 locally
(1000)	Channel binds to port 1000 locally
(1000,2000)	Channel binds to port in range 1000 - 2000 locally

When a channel is started the values specified for connection name (CONNAME) and local address (LOCLADDR) determine which IP stack is used for communication. The IP stack used is determined as follows:

- If the system has only an IPv4 stack configured, the IPv4 stack is always used. If a local address (LOCLADDR) or connection name (CONNAME) is specified as an IPv6 network address, an error is generated and the channel fails to start.
- If the system has only an IPv6 stack configured, the IPv6 stack is always used. If a local address (LOCLADDR) is specified as an IPv4 network address, an error is generated and the channel fails to start. On platforms supporting IPv6 mapped addressing, if a connection name (CONNAME) is specified as an IPv4 network address, the address is mapped to an IPv6 address. For example, xxx.xxx.xxx.xxx

is mapped to `::ffff:xxx.xxx.xxx.xxx`. The use of mapped addresses might require protocol translators. Avoid the use of mapped addresses where possible.

- If a local address (LOCLADDR) is specified as an IP address for a channel, the stack for that IP address is used. If the local address (LOCLADDR) is specified as a host name resolving to both IPv4 and IPv6 addresses, the connection name (CONNAME) determines which of the stacks is used. If both the local address (LOCLADDR) and connection name (CONNAME) are specified as host names resolving to both IPv4 and IPv6 addresses, the stack used is determined by the queue manager attribute IPADDRV.
- If the system has dual IPv4 and IPv6 stacks configured and a local address (LOCLADDR) is not specified for a channel, the connection name (CONNAME) specified for the channel determines which IP stack to use. If the connection name (CONNAME) is specified as a host name resolving to both IPv4 and IPv6 addresses, the stack used is determined by the queue manager attribute IPADDRV.

Multi On Multiplatforms, you can set a default local address value that is used for all sender channels that do not have a local address defined. The default value is defined by setting the MQ_LCLADDR environment variable prior to starting the queue manager. The format of the value matches that of MQSC attribute LOCLADDR.

Local addresses with cluster sender channels

Cluster sender channels always inherit the configuration of the corresponding cluster receiver channel as defined on the target queue manager. This is true even if there is a locally defined cluster sender channel of the same name, in which case the manual definition is only used for initial communication.

For this reason, it is not possible to depend on the LOCLADDR defined in the cluster receiver channel as it is likely that the IP address is not owned by the system where the cluster senders are created. For this reason, the LOCLADDR on the cluster receiver should not be used unless there is a reason to restrict only the ports but not the IP address for all potential cluster senders, and it is known that those ports are available on all systems where a cluster sender channel may be created.

If a cluster must use LOCLADDR to get the outbound communication channels to bind to a specific IP address, either use a Channel Auto-Definition Exit, or use the default LOCLADDR for the queue manager when possible. When using a channel exit, it forces the LOCLADDR value from the exit into any of the automatically defined CLUSSDR channels.

If using a non-default LOCLADDR for cluster sender channels through the use of an exit or a default value, any matching manually defined cluster sender channel, for example to a full repository queue manager, must also have the LOCLADDR value set to enable initial communication over the channel.

Note: If the operating system returns a bind error for the port supplied in LOCLADDR (or all ports, if a port range is supplied), the channel does not start; the system issues an error message.

LOCLADDR for AMQP channels

AMQP channels support a different format of LOCLADDR than other IBM MQ channels:

LOCLADDR (*ip-addr*)

LOCLADDR is the local communications address for the channel. Use this parameter if you want to force the client to use a particular IP address. LOCLADDR is also useful to force a channel to use an IPv4 or IPv6 address if a choice is available, or to use a particular network adapter on a system with multiple network adapters.

The maximum length of LOCLADDR is MQ_LOCAL_ADDRESS_LENGTH.

If you omit LOCLADDR, a local address is automatically allocated.

ip-addr

ip-addr is a single network address, specified in one of three forms:

IPv4 dotted decimal

For example 192.0.2.1

IPv6 hexadecimal notation
For example `2001:DB8:0:0:0:0:0:0`

Alphanumeric host name form
For example `WWW.EXAMPLE.COM`

If an IP address is entered, only the address format is validated. The IP address itself is not validated.

See Working with auto-defined cluster-sender channels for additional information.

LONGRTY (Long retry count)

This attribute specifies the maximum number of times that the channel is to try allocating a session to its partner.

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

This attribute is valid for the following channel types:

- Sender
- Server
- Cluster sender
- Cluster receiver

If the initial allocation attempt fails, the *short retry count* number is decremented and the channel retries the remaining number of times. If it still fails, it retries a *long retry count* number of times with an interval of *long retry interval* between each try. If it is still unsuccessful, the channel closes down. The channel must then be restarted with a command; it is not started automatically by the channel initiator.

z/OS On z/OS, a channel cannot enter retry if the maximum number of channels (**MAXCHL**) has been exceeded.

Multi On Multiplatforms, in order for retry to be attempted a channel initiator must be running. The channel initiator must be monitoring the initiation queue specified in the definition of the transmission queue that the channel is using.

If the channel initiator (on z/OS) or the channel (on Multiplatforms) is stopped while the channel is retrying, the *short retry count* and *long retry count* are reset when the channel initiator or the channel is restarted, or when a message is successfully put at the sender channel. However, if the channel initiator (on z/OS) or queue manager (on Multiplatforms) is shut down and restarted, the *short retry count* and *long retry count* are not reset. The channel retains the retry count values it had before the queue manager restart or the message being put.

Multi On Multiplatforms:

1. When a channel goes from RETRYING state to RUNNING state, the *short retry count* and *long retry count* are not reset immediately. They are reset only when the first message flows across the channel successfully after the channel went into RUNNING state, that is; when the local channel confirms the number of messages sent to the other end.
2. The *short retry count* and *long retry count* are reset when the channel is restarted.

LONGTMR (Long retry interval)

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

Channel attributes for MQSC keywords (M)

An alphabetical list of the channel attributes for MQSC keywords, starting with the letter *M*.

MAXINST (Maximum instances)

This attribute specifies the maximum number of simultaneous instances of a server-connection channel or AMQP channel that can be started.

Maximum instances of server-connection channel connections

For a server-connection channel, 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.

If the value is reduced so that it is less than the number of instances of the server-connection channel that are currently running, then the running channels are not affected. However, new instances are not able to start until sufficient existing ones have ceased to run.

Maximum instances of AMQP channel connections

For an AMQP channel, this attribute specifies the maximum number of simultaneous instances of an AMQP channel that can be started.

This attribute can be set from zero through 999 999 999. A value of zero indicates that no client connections are allowed on this channel. The default value is 999 999 999.

If a client attempts to connect, and the number of connected clients has reached MAXINST, the channel closes the connection with a close frame. The close frame contains the following message:

```
amqp:resource-limit-exceeded
```

If a client connects with an ID that is already connected (that is, it performs a client-takeover), the takeover will succeed regardless of whether the number of connected clients has reached MAXINST.

See Server-connection channel limits for additional information.

MAXINSTC (Maximum instances per client)

This attribute specifies the maximum number of simultaneous instances of a server-connection channel that can be started from a single client.

This attribute can be set from zero through 999 999 999. A value of zero indicates that no client connections are allowed on this channel. The default value is 999 999 999.

If the value is reduced so that it is less than the number of instances of the server-connection channel that are currently running from individual clients, then the running channels are not affected. However, new instances from those clients are not able to start until sufficient existing ones have ceased to run.

This attribute is valid only for server-connection channels.

See Server-connection channel limits for additional information.

MAXMSGL (Maximum message length)

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

Multi On Multiplatforms, specify a value greater than or equal to zero, and less than or equal to the maximum message length for the queue manager. See the MAXMSGL parameter of the ALTER QMGR command in ALTER QMGR for more information.

z/OS On IBM MQ for z/OS, specify a value greater than or equal to zero, and less than or equal to 104 857 600 bytes (that is, 100 MB).

Because various implementations of IBM MQ systems exist on different platforms, the size available for message processing might be limited in some applications. This number must reflect a size that your system can handle without stress. When a channel starts, the lower of the two numbers at each end of the channel is taken.

Note: You can use a maximum message size of 0 for the channel, which is taken to mean that the size is to be set to the local queue manager maximum value.

By adding the digital signature and key to the message, Advanced Message Security increases the length of the message.

This attribute is valid for all channel types.

MCANAME (Message channel agent name)

This attribute is reserved, if specified must be set only to blanks and has a maximum length is 20 characters.

MCATYPE (Message channel agent type)

This attribute can specify the message channel agent as a process or a thread.

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

Note: For channel types of sender, server, and requester, the default is `process`. For channel types of cluster-sender and cluster-receiver, the default is `thread`. These defaults can change during your installation.

If you specify `process` on the channel definition, a RUNMQCHL process is started. If you specify `thread`, the MCA runs on a thread of the AMQRMPPA process, or of the RUNMQCHI process if **MQNOREMPOOL** is specified. On the machine that receives the inbound allocates, the MCA runs as a thread if you use **RUNMQLSR**. It runs as a process if you use **inetd**.

z/OS On IBM MQ for z/OS, this attribute is supported only for channels with a channel type of cluster-receiver.

Multi On other platforms, this attribute is valid for channel types of:

- Sender
- Server
- Requester

- Cluster sender
- Cluster receiver

MCAUSER (Message channel agent user identifier)

This attribute is the user identifier (a string) to be used by the MCA for authorization to access IBM MQ resources.

Note: An alternative way of providing a user ID for a channel to run under is to use channel authentication records. With channel authentication records, different connections can use the same channel while using different credentials. If both MCAUSER on the channel is set and channel authentication records are used to apply to the same channel, the channel authentication records take precedence. The MCAUSER on the channel definition is only used if the channel authentication record uses USERSRC(CHANNEL).

This authorization includes (if PUT authority is DEF) putting the message to the destination queue for receiver or requester channels.

On IBM MQ for Windows, the user identifier can be domain-qualified by using the format, `user@domain`, where the `domain` must be either the Windows systems domain of the local system, or a trusted domain.

If this attribute is blank, the MCA uses its default user identifier. For more information, see DEFINE CHANNEL.

This attribute is valid for channel types of:

- Receiver
- Requester
- Server connection
- Cluster receiver

MODENAME (LU 6.2 mode name)

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 only for channel types of:

- Sender
- Server
- Requester
- Client connection
- Cluster sender
- Cluster receiver

MONCHL (Monitoring)

This attribute controls the collection of online Monitoring data.

Possible values are:

QMGR
The collection of Online Monitoring Data is inherited from the setting of the MONCHL attribute in the queue manager object. This value is the default value.

OFF
Online Monitoring Data collection for this channel is disabled.

LOW
A low ratio of data collection with a minimal effect on performance. However, the monitoring results shown might not be up to date.

MEDIUM
A moderate ratio of data collection with limited effect on the performance of the system.

HIGH
A high ratio of data collection with the possibility of an effect on performance. However, the monitoring results shown are the most current.

This attribute is valid for channel types of:

- Sender
- Server
- Receiver
- Requester
- Server connection
- Cluster sender
- Cluster receiver

For more information about monitoring data, see Displaying queue and channel monitoring data.

MRDATA (Message-retry exit user data)

his 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

MREXIT (Message-retry exit name)

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 “RCVEXIT (Receive exit name)” on page 116. However, you can specify only one message-retry exit.

This attribute is valid for channel types of:

- Receiver
- Requester
- Cluster receiver

MRRTY (Message retry count)

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

MRTMR (Message retry interval)

This attribute specifies the minimum interval of time in milliseconds that must pass before the channel can retry the MQPUT operation.

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

MSGDATA (Message exit user data)

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 “RCVDATA (Receive exit user data)” on page 116.

This attribute is valid for channel types of:

- Sender
- Server
- Receiver
- Requester
- Cluster sender
- Cluster receiver

MSGEXIT (Message exit name)

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 “RCVEXIT (Receive exit name)” on page 116.

This attribute is valid for channel types of:

- Sender
- Server
- Receiver
- Requester
- Cluster sender
- Cluster receiver

Channel attributes for MQSC keywords (N-R)

An alphabetical list of the channel attributes for MQSC keywords, starting with the letters *N* thru *R*.

NETPRTY (Network-connection priority)

The NETPRTY channel attribute specifies the priority for a CLUSRCVR channel. The value must be in the range 0-9, where 0 is the lowest priority and 9 is the highest.

Use the NETPRTY attribute to make one network the primary network, and another network the backup network. Given a set of equally ranked channels, clustering chooses the path with the highest priority when multiple paths are available.

A typical example of using the NETPRTY channel attribute is to differentiate between networks that have different costs or speeds and connect the same destinations.

Note: Specify this attribute on the cluster-receiver channel at the target queue manager. Any balancing you specify on the matching cluster-sender channel is likely to be ignored. See Cluster channels.

NPMSPEED (Nonpersistent message speed)

This attribute specifies the speed at which nonpersistent messages are to be sent.

Possible values are:

NORMAL
Nonpersistent messages on a channel are transferred within transactions.

FAST
Nonpersistent messages on a channel are not transferred within transactions.

The default is FAST. The advantage of this is that nonpersistent messages become available for retrieval far more quickly. The disadvantage is that because they are not part of a transaction, messages might be lost if there is a transmission failure or if the channel stops when the messages are in transit. See Safety of messages.

Notes:

1. If the active recovery logs for IBM MQ for z/OS are switching and archiving more frequently than expected, given that the messages being sent across a channel are non-persistent, setting NPMSPEED(FAST) on both the sending and receiving ends of the channel can minimize the SYSTEM.CHANNEL.SYNCQ updates.
2. If you are seeing high CPU usage relating to updates to the SYSTEM.CHANNEL.SYNCQ, setting NPMSPEED(FAST) can significantly reduce the CPU usage.

This attribute is valid for channel types of:

- Sender
- Server
- Receiver
- Requester
- Cluster sender
- Cluster receiver

PASSWORD (Password)

This attribute specifies a password that can be used by the MCA when attempting to initiate a secure LU 6.2 session with a remote MCA.

You can specify a password of maximum length 12 characters, although only the first 10 characters are used.

z/OS On IBM MQ for z/OS, this attribute is valid only for client connection channels.

Multi On other platforms, this attribute is valid for channel types of:

- Sender
- Server
- Requester
- Client connection
- Cluster sender

PORT (Port number)

Specify the port number that is used to connect the AMQP client.

The default port for AMQP 1.0 connections is 5672. If you are already using port 5672, you can specify a different port.

PUTAUT (PUT authority)

This attribute specifies the type of security processing to be carried out by the MCA.

Use this attribute to choose the type of security processing to be carried out by the MCA when executing:

- An MQPUT command to the destination queue (for message channels), or
- An MQI call (for MQI channels).

z/OS On z/OS, the user IDs that are checked, and how many user IDs are checked, depends on the setting of the MQADMIN RACF® class hlq.RESLEVEL profile. Depending on the level of access the user ID of the channel initiator has to hlq.RESLEVEL, zero, one or two user IDs are checked. To see how many user IDs are checked, see RESLEVEL and channel initiator connections. For more information about which user IDs are checked, see User IDs used by the channel initiator.

You can choose one of the following:

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

Multi On platforms other than z/OS, the user ID used to check open authority on the queue is that of the process or user running the MCA at the receiving end of the message channel.

z/OS On z/OS, both the user ID received from the network, and the user ID derived from MCAUSER might be used, depending on the number of user IDs that are to be checked.

The queues are opened with this user ID and the open option MQOO_SET_ALL_CONTEXT.

Context security (CTX)
The user ID from the context information associated with the message is used as an alternate user ID.

The *UserIdentifier* in the message descriptor is moved into the *AlternateUserId* field in the object descriptor. The queue is opened with the open options MQOO_SET_ALL_CONTEXT and MQOO_ALTERNATE_USER_AUTHORITY.

Multi On platforms other than z/OS, the user ID used to check open authority on the queue for MQOO_SET_ALL_CONTEXT and MQOO_ALTERNATE_USER_AUTHORITY is that of the process or user running the MCA at the receiving end of the message channel. The user ID used to check open authority on the queue for MQOO_OUTPUT is the *UserIdentifier* in the message descriptor.

z/OS On z/OS, the user ID received from the network or that derived from MCAUSER might be used, as well as the user ID from the context information in the message descriptor, depending on the number of user IDs that are to be checked.

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

z/OS **Only Message Channel Agent security (ONLYMCA)**
The user ID derived from MCAUSER is used.

Queues are opened with the open option MQOO_SET_ALL_CONTEXT.

This value only applies to z/OS.

z/OS **Alternate Message Channel Agent security (ALTMCA)**
The user ID from the context information (the *UserIdentifier* field) in the message descriptor might be used, as well as the user ID derived from MCAUSER, depending on the number of user IDs that are to be checked.

This value only applies to z/OS.

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

More information about security can be found here:

- Securing
- ALW Setting up security on AIX, Linux, and Windows
- IBM i Setting up security on IBM i
- z/OS Setting up security on z/OS

This attribute is valid for channel types of:

- Receiver
- Requester
- z/OS Server connection (z/OS only)
- Cluster receiver

QMNAME (Queue manager name)

his attribute specifies the name of the queue manager or queue manager group to which an IBM MQ MQI client application can request connection.

This attribute is valid for channel types of:

- Client connection

QSGDISP (Disposition)

z/OS

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.

RCVDATA (Receive exit user data)

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)
```

ALW In IBM MQ for UNIX systems, and Windows systems, the length of the string of exit names and strings of user data is limited to 500 characters.

IBM i In IBM MQ for IBM i, you can specify up to 10 exit names and the length of user data for each is limited to 32 characters.

z/OS In IBM MQ for z/OS, you can specify up to eight strings of user data each of length 32 characters.

This attribute is valid for all channel types.

RCVEXIT (Receive exit name)

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 it blank if no channel receive user exit is in effect.

The format and maximum length of this attribute depend on the platform:

- z/OS 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.
- IBM i On IBM i, it is of the form:

  ```
  libname/progname
  ```

  when specified in CL commands.

  When specified in IBM MQ Commands (MQSC) it has the form:

  ```
  progname libname
  ```

  where *progname* occupies the first 10 characters, and *libname* the second 10 characters (both blank-padded to the right if necessary). The maximum length of the string is 20 characters.
- Linux AIX On AIX and Linux, it is of the form:

  ```
  libraryname(functionname)
  ```

  The maximum length of the string is 40 characters.

- Windows 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.

z/OS 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.

- IBM i On IBM MQ for IBM i, you can list up to 10 exit names.
- z/OS On IBM MQ for z/OS, you can list up to eight exit names.

This attribute is valid for all channel types.

Channel attributes for MQSC keywords (S)

An alphabetical list of the channel attributes for MQSC keywords, starting with the letter *S*.

SCYDATA (Security exit user data)

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.

SCYEXIT (Security exit name)

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 RCVEXIT. However, you can only specify one security exit.

For more information about security exits, see Channel security exit programs.

This attribute is valid for all channel types.

SENDDATA (Send exit user data)

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 RCVDATA.

This attribute is valid for all channel types.

SENDEXIT (Send exit name)

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 RCVEXIT.

This attribute is valid for all channel types.

SEQWRAP (Sequence number wrap)

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

SHORTRTY (Short retry count)

This attribute specifies the maximum number of times that the channel is to try allocating a session to its partner.

The SHORTRTY attribute can be set from zero through 999 999 999.

If multiple IP addresses have been defined within the channel and reconnection is necessary, IBM MQ evaluates the channel definition and attempts to connect to each IP address in the order it is defined until either a successful connection is established or all addresses have been attempted.

In this case, SHORTRTY relates to how many total attempts the overall channel tries to reconnect, and not the individual IP addresses

If the initial allocation attempt fails, the *short retry count* is decremented and the channel retries the remaining number of times with an interval, defined in the **short retry interval** attribute, between each attempt. If it still fails, it retries *long retry count* number of times with an interval of *long retry interval* between each attempt. If it is still unsuccessful, the channel closes down.

z/OS On z/OS, a channel cannot enter retry if the maximum number of channels (**MAXCHL**) has been exceeded.

Multi On Multiplatforms, in order for retry to be attempted a channel initiator must be running. The channel initiator must be monitoring the initiation queue specified in the definition of the transmission queue that the channel is using.

If the channel initiator (on z/OS) or the channel (on Multiplatforms) is stopped while the channel is retrying, the *short retry count* and *long retry count* are reset when the channel initiator or the channel is restarted, or when a message is successfully put at the sender channel. However, if the channel initiator (on z/OS) or queue manager (on Multiplatforms) is shut down and restarted, the *short retry count* and *long retry count* are not reset. The channel retains the retry count values it had before the queue manager restart or the message being put.

Multi On Multiplatforms:

1. When a channel goes from RETRYING state to RUNNING state, the *short retry count* and *long retry count* are not reset immediately. They are reset only when the first message flows across the channel successfully after the channel went into RUNNING state, that is; when the local channel confirms the number of messages sent to the other end.
2. The *short retry count* and *long retry count* are reset when the channel is restarted.

This attribute is valid for the following channel types:

- Sender
- Server
- Cluster sender
- Cluster receiver

SHORTTMR (Short retry interval)

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.

If multiple IP addresses have been defined within the channel and reconnection is necessary, IBM MQ evaluates the channel definition and attempts to connect to each IP address in the order it is defined until either a successful connection is established or all addresses have been attempted.

In this case, SHORTTMR relates to how long the overall channel waits to restart the connection process, and not the individual IP addresses.

This attribute is valid for channel types of:

- Sender
- Server
- Cluster sender
- Cluster receiver

SPLPROT (Security policy protection)

This attribute specifies how a server-to-server Message Channel Agent should deal with message protection when AMS is active and an applicable policy exists.

This attribute can be set to:

PASSTHRU
On sender, server, receiver, and requester channels

REMOVE
On sender and server channels

ASPOLICY
On receiver and requester channels

This attribute is valid for channel types of:

- Sender
- Server
- Receiver
- Requester

SSLCAUTH (SSL client authentication)

The **SSLCAUTH** attribute specifies whether the channel needs to receive and authenticate a TLS certificate from a TLS client.

SSLCAUTH is an optional attribute. Possible values of this attribute are:

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

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

The default value is REQUIRED.

You can specify a value for **SSLCAUTH** on a non-TLS channel definition. That is, a channel definition on which the SSLCIPH attribute is missing or blank.

For more information about SSLCAUTH, see DEFINE CHANNEL (MQTT) and Securing.

The **SSLCAUTH** 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

SSLCIPH (SSL cipher specification)

The **SSLCIPH** attribute specifies an Alias or a single named CipherSpec for a TLS connection.

Every IBM MQ channel definition includes the **SSLCIPH** attribute. The value is a string with a maximum length of 32 characters.

The **SSLCIPH** attribute 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.

Notes:

- The **SSLCIPH** attribute can contain a blank value, meaning that you are not using TLS. If one end of the channel has a blank **SSLCIPH** attribute, the other end of the channel must also have a blank SSLCIPH attribute.

  Multi V 9.3.0 If SecureCommsOnly is enabled, plain text communication is not supported and the channel fails to start.
- Alternatively, if **SSLCIPH** contains a nonblank value, the value can be either an Alias or a named CipherSpec. The channels negotiate the strongest CipherSpec supported by both ends of the channel.
- A fully-managed .NET client can specify the special value *NEGOTIATE. This option allows the channel to select the most recent protocol version supported by the .NET framework, and negotiate a CipherSpec that the server supports.

The **SSLCIPH** attribute 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.

SSLPEER (SSL peer)

The **SSLPEER** attribute is used to check the Distinguished Name (DN) of the certificate from the peer queue manager or client at the other end of an IBM MQ channel.

Note: An alternative way of restricting connections into channels by matching against the TLS Subject Distinguished Name, is to use channel authentication records. With channel authentication records, different TLS Subject Distinguished Name patterns can be applied to the same channel. If both **SSLPEER** on the channel and a channel authentication record are used to apply to the same channel, the inbound certificate must match both patterns in order to connect.

If the DN received from the peer does not match the **SSLPEER** value, the channel does not start.

SSLPEER is an optional attribute. If a value is not specified, the peer DN is not checked when the channel is started.

The maximum length of the **SSLPEER** attribute depends on the platform:

- z/OS On z/OS, the maximum length of the attribute is 256 bytes.
- Multi On all other platforms, it is 1024 bytes.

Channel authentication records provide greater flexibility when using **SSLPEER** and support a maximum length of 1024 bytes on all platforms.

The checking of **SSLPEER** attribute values also depends on the platform:

- z/OS On z/OS, the attribute values that are 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.
- Multi On platforms other than z/OS that support **SSLPEER**, the validity of the string is checked when it is first entered.

You can specify a value for **SSLPEER** on a non-TLS channel definition, one on which the SSLCIPH attribute is missing or blank. You can use this to temporarily disable TLS for debugging without having to clear and later re-input the TLS parameters.

The **SSLPEER** attribute is valid for all channel types.

For more information about using **SSLPEER**, see SET CHLAUTH, Securing, and Channel authentication records.

STATCHL (channel statistics)

This attribute controls the collection of statistics data for channels.

The possible values are:

QMGR
Statistics data collection for this channel is based upon the setting of the queue manager attribute STATCHL. This value is the default value.

OFF
Statistics data collection for this channel is disabled.

LOW
Statistics data collection for this channel is enabled with a low ratio of data collection.

MEDIUM
Statistics data collection for this channel is enabled with a moderate ratio of data collection.

HIGH
Statistics data collection for this channel is enabled with a high ratio of data collection.

For more information about channel statistics, see Monitoring reference.

z/OS On z/OS systems, enabling this parameter simply turns on statistics data collection, regardless of the value you select. Specifying `LOW`, `MEDIUM`, or `HIGH` makes no difference to your results. This parameter must be enabled in order to collect channel accounting records.

This attribute is valid for channel types of:

- Sender
- Server
- Receiver
- Requester
- Cluster sender
- Cluster receiver

Channel attributes for MQSC keywords (T-Z)

An alphabetical list of the channel attributes for MQSC keywords, starting with the letters *T* thru *Z*.

TPNAME (LU 6.2 transaction program name)

This attribute is for use with LU 6.2 connections. It is the name, or generic name, of the transaction program (MCA) to be run at the far end of the link.

When using side information for SNA communications, the transaction program name is defined in the CPI-C Communications Side Object or APPC side information and this attribute must be left blank. Otherwise, this name is required by sender channels and requester channels.

The name can be up to 64 characters long.

The name must be set to the SNA transaction program name, unless the CONNAME contains a side-object name in which case it must be set to blanks. The actual name is taken instead from the CPI-C Communications Side Object, or the APPC side information data set.

This information is set in different ways on different platforms; see Configuring distributed queuing for more information about setting up communication for your platform.

This attribute is valid for channel types of:

- Sender
- Server
- Requester
- Client connection
- Cluster sender
- Cluster receiver

TPROOT (Topic root)

This attribute specifies the topic root for an AMQP channel.

You can use the TPROOT attribute to specify a topic root for an AMQP channel. Using this attribute ensures that an MQ Light application, when deployed to a queue manager, does not publish or subscribe to messages to or from areas of the topic tree that are being used by other applications.

The default value for TPROOT is SYSTEM.BASE.TOPIC. With this value, the topic string an AMQP client uses to publish or subscribe has no prefix, and the client can exchange messages with other MQ pub/sub applications. To have AMQP clients publish and subscribe under a topic prefix, first create an MQ topic object with a topic string set to the prefix you want, then change the value of the AMQP channel TPROOT attribute to the name of the MQ topic object you created. The following example shows the topic root being set to APPGROUP1.BASE.TOPIC for AMQP channel MYAMQP:

```
DEFINE CHANNEL(MYAMQP) CHLTYPE(AMQP) TPROOT(APPGROUP1.BASE.TOPIC) PORT(5673)
```

Note: If the TPROOT attribute value, or the topic string that underpins it, is changed, existing AMQP topics and their messages might be orphaned.

TRPTYPE (Transport type)

This attribute specifies the transport type to be used.

The possible values are:

Table 58. Transport type options	
Value	**Transport type**
LU62	LU 6.2
TCP	TCP/IP
NETBIOS	NetBIOS "1" on page 123
SPX	SPX "1" on page 123

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.

USECLTID (Use client ID)

Specify whether the client ID is used for connection on an AMQP channel. Set to Yes or No.

USEDLQ (Use dead-Letter queue)

This attribute determines whether the dead-letter queue (or undelivered message queue) is used when messages cannot be delivered by channels.

Possible values are:

NO
Messages that cannot be delivered by a channel are treated as a failure. The channel either discards these messages, or the channel ends, in accordance with the setting of NPMSPEED.

YES (default)
If the queue manager DEADQ attribute provides the name of a dead-letter queue, then it is used, otherwise the behavior is as for NO.

USERID (User ID)

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.

On the receiving end, if passwords are kept in encrypted format and the LU 6.2 software is using a different encryption method, an attempt to start the channel fails with invalid security details. You can avoid this failure by modifying the receiving SNA configuration to either:

- Turn off password substitution, or
- Define a security user ID and password.

On IBM MQ for z/OS, this attribute is valid only for client connection channels.

On other platforms, this attribute is valid for channel types of:

- Sender
- Server
- Requester

- Client connection
- Cluster sender

XMITQ (Transmission queue name)

This attribute specifies the name of the transmission queue from which messages are retrieved.

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 required for channels of type sender or server and is not valid for other channel types.

IBM MQ cluster commands and attributes

There are MQSC and PCF cluster commands that you can use to refresh or reset a cluster, or to display, resume or suspend a cluster queue manager. In addition, the MQSC and PCF commands that define channels, queues, and queue managers have attributes that apply to clusters. Some of these attributes are used by the cluster workload management algorithm.

MQSC commands

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:

- IBM i On IBM MQ for IBM i, you run MQSC commands interactively from option 26 of **WRKMQM**. You can also use CL commands or you can store MQSC commands in a file and use the **STRMQMMQSC** CL command.
- z/OS On z/OS you can use the COMMAND function of the **CSQUTIL** utility, the operations and control panels, or the z/OS console.
- On all other platforms, you can store the commands in a file and use **runmqsc**.

In an MQSC command, a cluster name that is specified using the CLUSTER attribute can be up to 48 characters long.

A list of cluster names that is specified using the CLUSNL attribute can contain up to 256 names. To create a cluster namelist, use the DEFINE NAMELIST command.

IBM MQ Explorer

The IBM MQ Explorer GUI can administer a cluster with repository queue managers on IBM WebSphere MQ for z/OS 6.0 or later. You do not need to nominate an additional repository on a separate system. For earlier versions of IBM MQ for z/OS, the IBM MQ Explorer cannot administer a cluster with repository queue managers. You must therefore nominate an additional repository on a system that the IBM MQ Explorer can administer.

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

Using the IBM MQ Explorer, you can view cluster queues and inquire about the status of cluster-sender and cluster-receiver channels. IBM MQ Explorer includes two wizards, which you can use to guide you through the following tasks:

- Create a cluster
- Join an independent queue manager to a cluster

PCF equivalents of MQSC commands specifically to work with clusters

MQSC command	Equivalent PCF command
DISPLAY CLUSQMGR	MQCMD_INQUIRE_CLUSTER_Q_MGR
REFRESH CLUSTER	MQCMD_REFRESH_CLUSTER
RESET CLUSTER	MQCMD_RESET_CLUSTER
RESUME QMGR	MQCMD_RESUME_Q_MGR_CLUSTER
SUSPEND QMGR	MQCMD_SUSPEND_Q_MGR_CLUSTER

Related information

Clustering: Using REFRESH CLUSTER best practices

Cluster attributes available on channel definition commands

Cluster attributes that can be specified on channel definition commands.

The DEFINE CHANNEL, ALTER CHANNEL, and DISPLAY CHANNEL commands have two specific CHLTYPE parameters for clusters: CLUSRCVR and CLUSSDR. To define a cluster-receiver channel you use the DEFINE CHANNEL command, specifying CHLTYPE(CLUSRCVR). Many attributes on a cluster-receiver channel definition are the same as the attributes on a receiver or sender-channel definition. To define a cluster-sender channel you use the DEFINE CHANNEL command, specifying CHLTYPE(CLUSSDR), and many of the same attributes as you use to define a sender-channel.

It is no longer necessary to specify the name of the full repository queue manager when you define a cluster-sender channel. If you know the naming convention used for channels in your cluster, you can make a CLUSSDR definition using the +QMNAME+ construction. The +QMNAME+ construction is not supported on z/OS. After connection, IBM MQ changes the name of the channel and substitutes the correct full repository queue manager name in place of +QMNAME+. The resulting channel name is truncated to 20 characters.

For more information on naming conventions, see Cluster naming conventions.

The technique works only if your convention for naming channels includes the name of the queue manager. For example, you define a full repository queue manager called QM1 in a cluster called CLUSTER1 with a cluster-receiver channel called CLUSTER1.QM1.ALPHA. Every other queue manager can define a cluster-sender channel to this queue manager using the channel name, CLUSTER1.+QMNAME+.ALPHA.

If you use the same naming convention for all your channels, be aware that only one +QMNAME+ definition can exist at one time.

The following attributes on the DEFINE CHANNEL and ALTER CHANNEL commands are specific to cluster channels:

CLUSTER
The CLUSTER attribute specifies the name of the cluster with which this channel is associated. Alternatively use the CLUSNL attribute.

CLUSNL
The CLUSNL attribute specifies a namelist of cluster names.

NETPRTY
Cluster-receivers only.

The NETPRTY attribute specifies a network priority for the channel. NETPRTY helps the workload management routines. If there is more than one possible route to a destination, the workload management routine selects the one with the highest priority.

CLWLPRTY
The CLWLPRTY parameter applies a priority factor to channels to the same destination for workload management purposes. This parameter specifies the priority of the channel for the purposes of cluster

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

CLWLRANK
The CLWLRANK parameter applies a ranking factor to a channel for workload management purposes. This parameter specifies the rank of a channel for the purposes of cluster workload distribution. The value must be in the range zero through 9, where zero is the lowest rank and 9 is the highest.

CLWLWGHT
The CLWLWGHT parameter applies a weighting factor to a channel for workload management purposes. CLWLWGHT weights the channel so that the proportion of messages sent down that channel can be controlled. The cluster workload algorithm uses CLWLWGHT to bias the destination choice so that more messages can be sent over a particular channel. By default all channel weight attributes are the same default value. The weight attribute allows you to allocate a channel on a powerful UNIX machine a larger weight than another channel on small desktop PC. The greater weight means that the cluster workload algorithm selects the UNIX machine more frequently than the PC as the destination for messages.

CONNAME
The CONNAME specified on a cluster-receiver channel definition is used throughout the cluster to identify the network address of the queue manager. Take care to select a value for the CONNAME parameter that resolves throughout your IBM MQ cluster. Do not use a generic name. Remember that the value specified on the cluster-receiver channel takes precedence over any value specified in a corresponding cluster-sender channel.

These attributes on the DEFINE CHANNEL command and ALTER CHANNEL command also apply to the DISPLAY CHANNEL command.

Note: Auto-defined cluster-sender channels take their attributes from the corresponding cluster-receiver channel definition on the receiving queue manager. Even if there is a manually defined cluster-sender channel, its attributes are automatically modified to ensure that they match the attributes on the corresponding cluster-receiver definition. Beware that you can, for example, define a CLUSRCVR without specifying a port number in the CONNAME parameter, while manually defining a CLUSSDR that does specify a port number. When the auto-defined CLUSSDR replaces the manually defined one, the port number (taken from the CLUSRCVR) becomes blank. The default port number would be used and the channel would fail.

Note: The DISPLAY CHANNEL command does not display auto-defined channels. However, you can use the DISPLAY CLUSQMGR command to examine the attributes of auto-defined cluster-sender channels.

Use the DISPLAY CHSTATUS command to display the status of a cluster-sender or cluster-receiver channel. This command gives the status of both manually defined channels and auto-defined channels.

The equivalent PCFs are MQCMD_CHANGE_CHANNEL, MQCMD_COPY_CHANNEL, MQCMD_CREATE_CHANNEL, and MQCMD_INQUIRE_CHANNEL.

Omitting the CONNAME value on a CLUSRCVR definition

In some circumstances you can omit the CONNAME value on a CLUSRCVR definition. You must not omit the CONNAME value on z/OS.

Multi On Multiplatforms, the TCP/IP connection name parameter of a cluster-receiver channel is optional. If you leave the connection name blank, IBM MQ generates a connection name for you, assuming the default port and using the current IP address of the system. You can override the default port number, but still use the current IP address of the system. For each connection name leave the IP name blank, and provide the port number in parentheses; for example:

```
(1415)
```

The generated **CONNAME** is always in the dotted decimal (IPv4) or hexadecimal (IPv6) form, rather than in the form of an alphanumeric DNS host name.

This facility is useful when you have machines using Dynamic Host Configuration Protocol (DHCP). If you do not supply a value for the CONNAME on a CLUSRCVR channel, you do not need to change the CLUSRCVR definition. DHCP allocates you a new IP address.

If you specify a blank for the CONNAME on the CLUSRCVR definition, IBM MQ generates a CONNAME from the IP address of the system. Only the generated CONNAME is stored in the repositories. Other queue managers in the cluster do not know that the CONNAME was originally blank.

If you issue the DISPLAY CLUSQMGR command you see the generated CONNAME. However, if you issue the DISPLAY CHANNEL command from the local queue manager, you see that the CONNAME is blank.

If the queue manager is stopped and restarted with a different IP address, because of DHCP, IBM MQ regenerates the CONNAME and updates the repositories accordingly.

Related concepts

Workload balancing in clusters
If a cluster contains more than one instance of the same queue, IBM MQ selects a queue manager to route a message to. It uses the cluster workload management algorithm, and a number of cluster workload-specific attributes, to determine the best queue manager to use.

z/OS Asynchronous behavior of CLUSTER commands on z/OS
The command issuer of a cluster command on z/OS receives confirmation a command has been sent, but not that it has completed successfully.

Related reference

Cluster attributes available on queue definition commands
Cluster attributes that can be specified on the queue definition commands.

Cluster attributes available on queue manager definition commands
Cluster attributes that can be specified on queue manager 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 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.

"Cluster workload balancing - channel attributes" on page 138
An alphabetical list of the channel attributes used in cluster workload balancing.

Cluster attributes available on queue definition commands

Cluster attributes that can be specified on the queue definition commands.

The DEFINE QLOCAL, DEFINE QREMOTE, and DEFINE QALIAS commands

The cluster attributes on the DEFINE QLOCAL, DEFINE QREMOTE, and DEFINE QALIAS commands, and the three equivalent ALTER commands, are:

CLUSTER
Specifies the name of the cluster to which the queue belongs.

CLUSNL
Specifies a namelist of cluster names.

DEFBIND
Specifies the binding to be used when an application specifies MQOO_BIND_AS_Q_DEF on the MQOPEN call. The options for this attribute are:

- Specify DEFBIND(OPEN) to bind the queue handle to a specific instance of the cluster queue when the queue is opened. DEFBIND(OPEN) is the default for this attribute.
- Specify DEFBIND(NOTFIXED) so that the queue handle is not bound to any instance of the cluster queue.
- Specify DEFBIND(GROUP) to allow an application to request that a group of messages are all allocated to the same destination instance.

When multiple queues with the same name are advertised in a Queue Manager Cluster, applications can choose whether to send all messages from this application to a single instance (MQOO_BIND_ON_OPEN), to allow the workload management algorithm to select the most suitable destination on a per message basis (MQOO_BIND_NOT_FIXED), or allow an application to request that a 'group' of messages be all allocated to the same destination instance (MQOO_BIND_ON_GROUP). The workload balancing is re-driven between groups of messages (without requiring an MQCLOSE and MQOPEN of the queue).

When you specify DEFBIND on a queue definition, the queue is defined with one of the attributes, MQBND_BIND_ON_OPEN, MQBND_BIND_NOT_FIXED, or MQBND_BIND_ON_GROUP. Either MQBND_BIND_ON_OPEN or MQBND_BIND_ON_GROUP must be specified when using groups with clusters.

You should set the DEFBIND attribute to the same value on all instances of the same cluster queue.

CLWLRANK
Applies a ranking factor to a queue for workload management purposes. CLWLRANK parameter is not supported on model queues. The cluster workload algorithm selects a destination queue with the highest rank. By default CLWLRANK for all queues is set to zero.

If the final destination is a queue manager on a different cluster, you can set the rank of any intermediate gateway queue managers at the intersection of neighboring clusters. With the intermediate queue managers ranked, the cluster workload algorithm correctly selects a destination queue manager nearer the final destination.

The same logic applies to alias queues. The rank selection is made before the channel status is checked, and therefore even non-accessible queue managers are available for selection. This has the effect of allowing a message to be routed through a network, rather than having it select between two possible destinations (as the priority would). So, if a channel is not started to the place where the rank has indicated, the message is not routed to the next highest rank, but waits until a channel is available to that destination (the message is held on the transmit queue).

CLWLPRTY
Applies a priority factor to a queue for workload management purposes. The cluster workload algorithm selects a destination queue with the highest priority. By default priority for all queues is set to zero.

If there are two possible destination queues, you can use this attribute to make one destination failover to the other destination. The priority selection is made after the channel status is checked. All messages are sent to the queue with the highest priority unless the status of the channel to that destination is not as favorable as the status of channels to other destinations. This means that only the most accessible destinations are available for selection. This has the effect of prioritizing between multiple destinations that are all available.

CLWLUSEQ
Specifies the behavior of an MQPUT operation for a queue. This parameter specifies the behavior of an MQPUT operation when the target queue has a local instance and at least one remote cluster instance (except where the MQPUT originates from a cluster channel). This parameter is only valid for local queues.

Possible values are: QMGR (the behavior is as specified by the CLWLUSEQ parameter of the queue manager definition), ANY (the queue manager treats the local queue as another instance of the cluster queue, for the purposes of workload distribution), LOCAL (the local queue is the only target of the

MQPUT operation, providing the local queue is put enabled). The MQPUT behavior depends upon the cluster workload management algorithm.

The DISPLAY QUEUE and DISPLAY QCLUSTER commands

The attributes on the DEFINE QLOCAL, DEFINE QREMOTE, and DEFINE QALIAS commands also apply to the DISPLAY QUEUE command.

To display information about cluster queues, specify a queue type of QCLUSTER or the keyword CLUSINFO on the DISPLAY QUEUE command, or use the command DISPLAY QCLUSTER.

The DISPLAY QUEUE or DISPLAY QCLUSTER command returns the name of the queue manager that hosts the queue (or the names of all queue managers if there is more than one instance of the queue). It also returns the system name for each queue manager that hosts the queue, the queue type represented, and the date and time at which the definition became available to the local queue manager. This information is returned using the CLUSQMGR, QMID, CLUSQT, CLUSDATE, and CLUSTIME attributes.

The system name for the queue manager (QMID), is a unique, system-generated name for the queue manager.

You can define a cluster queue that is also a shared queue. For example. on z/OS you can define:

```
DEFINE QLOCAL(MYQUEUE) CLUSTER(MYCLUSTER) QSGDISP(SHARED) CFSTRUCT(STRUCTURE)
```

The equivalent PCFs are MQCMD_CHANGE_Q, MQCMD_COPY_Q, MQCMD_CREATE_Q, and MQCMD_INQUIRE_Q.

Related concepts

Workload balancing in clusters
If a cluster contains more than one instance of the same queue, IBM MQ selects a queue manager to route a message to. It uses the cluster workload management algorithm, and a number of cluster workload-specific attributes, to determine the best queue manager to use.

z/OS Asynchronous behavior of CLUSTER commands on z/OS
The command issuer of a cluster command on z/OS receives confirmation a command has been sent, but not that it has completed successfully.

Related reference

Cluster attributes available on channel definition commands
Cluster attributes that can be specified on channel definition commands.

Cluster attributes available on queue manager definition commands
Cluster attributes that can be specified on queue manager 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 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.

“Cluster workload balancing - queue attributes” on page 140

An alphabetical list of queue attributes used in cluster workload balancing.

Cluster attributes available on 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 specify the attribute REPOSNL(*namelist*) on the **ALTER QMGR** command:

```
DEFINE NAMELIST(CLUSTERLIST)
       DESCR('List of clusters whose repositories I host')
       NAMES(CLUS1, CLUS2, CLUS3)
 ALTER QMGR REPOSNL(CLUSTERLIST)
```

You can provide additional cluster attributes on the ALTER QMGR command

CLWLEXIT(*name*)
Specifies the name of a user exit to be called when a message is put to a cluster queue.

CLWLDATA(*data*)
Specifies the data to be passed to the cluster workload user exit.

CLWLLEN(*length*)
Specifies the maximum amount of message data to be passed to the cluster workload user exit

CLWLMRUC(*channels*)
Specifies the maximum number of outbound cluster channels.

CLWLMRUC is a local queue manager attribute that is not propagated around the cluster. It is made available to cluster workload exits and the cluster workload algorithm that chooses the destination for messages.

CLWLUSEQ(LOCAL|ANY)
Specifies the behavior of MQPUT when the target queue has both a local instance and at least one remote cluster instance. If the put originates from a cluster channel, this attribute does not apply. It is possible to specify CLWLUSEQ as both a queue attribute and a queue manager attribute.

If you specify ANY, both the local queue and the remote queues are possible targets of the MQPUT.

If you specify LOCAL, the local queue is the only target of the MQPUT.

The equivalent PCFs are MQCMD_CHANGE_Q_MGR and MQCMD_INQUIRE_Q_MGR.

Related concepts

Workload balancing in clusters
If a cluster contains more than one instance of the same queue, IBM MQ selects a queue manager to route a message to. It uses the cluster workload management algorithm, and a number of cluster workload-specific attributes, to determine the best queue manager to use.

z/OS Asynchronous behavior of CLUSTER commands on z/OS
The command issuer of a cluster command on z/OS receives confirmation a command has been sent, but not that it has completed successfully.

Related reference

Cluster attributes available on channel definition commands
Cluster attributes that can be specified on channel definition commands.

Cluster attributes available on 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 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.

"Cluster workload balancing - queue manager attributes" on page 141
An alphabetical list of queue manager attributes used in cluster workload balancing.

DISPLAY CLUSQMGR

Use the DISPLAY CLUSQMGR command to display cluster information about queue managers in a cluster.

If you issue this command from a queue manager with a full repository, the information returned applies to every queue manager in the cluster. Otherwise the information returned applies only to the queue managers in which it has an interest. That is, every queue manager to which it has tried to send a message and every queue manager that holds a full repository.

The information includes most channel attributes that apply to cluster-sender and cluster-receiver channels. In addition, the following attributes can be displayed:

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

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

CLUSTER
What clusters the queue manager is in.

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

DEFTYPE
How the queue manager was defined. DEFTYPE can be one of the following values:

CLUSSDR
A cluster sender-channel has been administratively defined on the local queue manager but not yet recognized by the target queue manager. To be in this state the local queue manager has defined a manual cluster-sender channel but the receiving queue manager has not accepted the cluster information. This may be due to the channel never having been established due to availability or to an error in the cluster-sender configuration, for example a mismatch in the CLUSTER property between the sender and receiver definitions. This is a transitory condition or error state and should be investigated.

CLUSSDRA
This value represents an automatically discovered cluster queue manager, no cluster-sender channel is defined locally. This is the DEFTYPE for cluster queue managers for which the local queue manager has no local configuration but has been informed of. For example

- If the local queue manager is a full repository queue manager it should be the DEFTYPE value for all partial repository queue managers in the cluster.
- If the local queue manager is a partial repository, this could be the host of a cluster queue that is being used from this local queue manager or from a second full repository queue manager that this queue manager has been told to work with.

If the DEFTYPE value is CLUSSDRA and the local and remote queue managers are both full repositories for the named cluster, the configuration is not correct as a locally defined cluster-sender channel must be defined to convert this to a DEFTYPE of CLUSSDRB.

CLUSSDRB
A cluster sender-channel has been administratively defined on the local queue manager and accepted as a valid cluster channel by the target queue manager. This is the expected DEFTYPE of a partial repository queue manager's manually configured full repository queue manager. It should also be the DEFTYPE of any CLUSQMGR from one full repository to another full repository in the cluster. Manual cluster-sender channels should not be configured to partial repositories or from a partial repository queue manager to more than one full repository. If a DEFTYPE of CLUSSDRB is seen in either of these situations it should be investigated and corrected.

CLUSRCVR
Administratively defined as a cluster-receiver channel on the local queue manager. This represents the local queue manager in the cluster.

Note: To identify which CLUSQMGRs are full repository queue managers for the cluster, see the QMTYPE property.

For more information on defining cluster channels, see Cluster channels.

QMTYPE
Whether it holds a full repository or only a partial repository.

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

SUSPEND
Whether the queue manager is suspended.

VERSION
The version of the IBM MQ installation that the cluster queue manager is associated with.

The version has the format VVRRMMFF:

- VV: Version
- RR: Release
- MM: Maintenance level
- FF: Fix level

XMITQ
The cluster transmission queue used by the queue manager.

See also the DISPLAY QCLUSTER command. This is briefly described in DISPLAY QUEUE and in the DISPLAY QUEUE and DISPLAY QCLUSTER commands section of “Cluster attributes available on queue definition commands” on page 127. For examples of using DISPLAY QCLUSTER, search the information set for "DISPLAY QCLUSTER" and "DIS QCLUSTER".

Related concepts

Workload balancing in clusters
If a cluster contains more than one instance of the same queue, IBM MQ selects a queue manager to route a message to. It uses the cluster workload management algorithm, and a number of cluster workload-specific attributes, to determine the best queue manager to use.

z/OS Asynchronous behavior of CLUSTER commands on z/OS
The command issuer of a cluster command on z/OS receives confirmation a command has been sent, but not that it has completed successfully.

Related reference

Cluster attributes available on channel definition commands
Cluster attributes that can be specified on channel definition commands.

Cluster attributes available on queue definition commands
Cluster attributes that can be specified on the queue definition commands.

Cluster attributes available on queue manager definition commands

Cluster attributes that can be specified on queue manager definition commands.

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.

MQSC command **DISPLAY CLUSQMGR**

REFRESH CLUSTER

Issue the REFRESH CLUSTER command from a queue manager to discard all locally held information about a cluster. You are unlikely to need to use this command, except in exceptional circumstances.

There are three forms of this command:

REFRESH CLUSTER(clustername) REPOS(NO)
The default. The queue manager retains knowledge of all locally defined cluster queue manager and cluster queues and all cluster queue managers that are full repositories. In addition, if the queue manager is a full repository for the cluster it also retains knowledge of the other cluster queue managers in the cluster. Everything else is removed from the local copy of the repository and rebuilt from the other full repositories in the cluster. Cluster channels are not stopped if REPOS(NO) is used. A full repository uses its CLUSSDR channels to inform the rest of the cluster that it has completed its refresh.

REFRESH CLUSTER(clustername) REPOS(YES)
In addition to the default behavior, objects representing full repository cluster queue managers are also refreshed. It is not valid to use this option if the queue manager is a full repository, if used the command will fail with an error AMQ9406/CSQX406E logged. If it is a full repository, you must first alter it so that it is not a full repository for the cluster in question. The full repository location is recovered from the manually defined CLUSSDR definitions. After refreshing with REPOS(YES) has been issued the queue manager can be altered so that it is once again a full repository, if required.

REFRESH CLUSTER(*)
Refreshes the queue manager in all the clusters it is a member of. If used with REPOS(YES) REFRESH CLUSTER(*) has the additional effect of forcing the queue manager to restart its search for full repositories from the information in the local CLUSSDR definitions. The search takes place even if the CLUSSDR channel connects the queue manager to several clusters.

Note: For large clusters, use of the **REFRESH CLUSTER** command can be disruptive to the cluster while it is in progress, and again at 27 day intervals thereafter when the cluster objects automatically send status updates to all interested queue managers. See Refreshing in a large cluster can affect performance and availability of the cluster.

Related concepts

Workload balancing in clusters
If a cluster contains more than one instance of the same queue, IBM MQ selects a queue manager to route a message to. It uses the cluster workload management algorithm, and a number of cluster workload-specific attributes, to determine the best queue manager to use.

z/OS Asynchronous behavior of CLUSTER commands on z/OS
The command issuer of a cluster command on z/OS receives confirmation a command has been sent, but not that it has completed successfully.

Related reference

Cluster attributes available on channel definition commands

Cluster attributes that can be specified on channel definition commands.

Cluster attributes available on queue definition commands
Cluster attributes that can be specified on the queue definition commands.

Cluster attributes available on queue manager definition commands
Cluster attributes that can be specified on queue manager definition commands.

DISPLAY CLUSQMGR
Use the DISPLAY CLUSQMGR command to display cluster information about queue managers in a cluster.

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.

Related information

Clustering: Using REFRESH CLUSTER best practices

RESET CLUSTER: Forcibly removing a queue manager from a cluster

Use the **RESET CLUSTER** command to forcibly remove a queue manager from a cluster in exceptional circumstances.

You are unlikely to need to use this command, except in exceptional circumstances.

You can issue the **RESET CLUSTER** command only from full repository queue managers. The command takes two forms, depending on whether you reference the queue manager by name or identifier.

1.
```
RESET CLUSTER( clustername
) QMNAME( qmname ) ACTION(FORCEREMOVE) QUEUES(NO)
```

2.
```
RESET CLUSTER( clustername
) QMID( qmid ) ACTION(FORCEREMOVE) QUEUES(NO)
```

You cannot specify both QMNAME and QMID. If you use QMNAME, and there is more than one queue manager in the cluster with that name, the command is not run. Use QMID instead of QMNAME to ensure the **RESET CLUSTER** command is run.

Specifying QUEUES(NO) on a **RESET CLUSTER** command is the default. Specifying QUEUES(YES) removes references to cluster queues owned by the queue manager from the cluster. The references are removed in addition to removing the queue manager from the cluster itself.

The references are removed even if the cluster queue manager is not visible in the cluster; perhaps because it was previously forcibly removed, without the QUEUES option.

You might use the **RESET CLUSTER** command if, for example, a queue manager has been deleted but still has cluster-receiver channels defined to the cluster. Instead of waiting for IBM MQ to remove these definitions (which it does automatically) you can issue the **RESET CLUSTER** command to tidy up sooner. All other queue managers in the cluster are then informed that the queue manager is no longer available.

If a queue manager is temporarily damaged, you might want to tell the other queue managers in the cluster before they try to send it messages. **RESET CLUSTER** removes the damaged queue manager. Later, when the damaged queue manager is working again, use the **REFRESH CLUSTER** command to reverse the effect of **RESET CLUSTER** and return the queue manager to the cluster.If the queue manager is in a publish/subscribe cluster, you then need to reinstate any required proxy subscriptions. See REFRESH CLUSTER considerations for publish/subscribe clusters.

Note: For large clusters, use of the **REFRESH CLUSTER** command can be disruptive to the cluster while it is in progress, and again at 27 day intervals thereafter when the cluster objects automatically send status

updates to all interested queue managers. See Refreshing in a large cluster can affect performance and availability of the cluster.

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

Important: If the auto-defined channel to be removed is in-doubt, **RESET CLUSTER** does not immediately remove that channel. In this situation you need to issue a RESOLVE CHANNEL command, prior to the **RESET CLUSTER** command.

You are unlikely to need this command in normal circumstances. IBM Support might advise you to issue the command to tidy up the cluster information held by cluster queue managers. Do not use this command as a short cut to removing a queue manager from a cluster. The correct way to remove a queue manager from a cluster is described in Removing a queue manager from a cluster.

Because repositories retain information for only 90 days, after that time a queue manager that was forcibly removed can reconnect to a cluster. It reconnects automatically, unless it has been deleted. If you want to prevent a queue manager from rejoining a cluster, you need to take appropriate security measures.

All cluster commands, except **DISPLAY CLUSQMGR**, work asynchronously. Commands that change object attributes involving clustering update the object and send a request to the repository processor. Commands for working with clusters are checked for syntax, and a request is sent to the repository processor.

The requests sent to the repository processor are processed asynchronously, along with cluster requests received from other members of the cluster. Processing might take a considerable time if they have to be propagated around the whole cluster to determine if they are successful or not.

Related concepts

Workload balancing in clusters
If a cluster contains more than one instance of the same queue, IBM MQ selects a queue manager to route a message to. It uses the cluster workload management algorithm, and a number of cluster workload-specific attributes, to determine the best queue manager to use.

z/OS Asynchronous behavior of CLUSTER commands on z/OS
The command issuer of a cluster command on z/OS receives confirmation a command has been sent, but not that it has completed successfully.

Related reference

Cluster attributes available on channel definition commands
Cluster attributes that can be specified on channel definition commands.

Cluster attributes available on queue definition commands
Cluster attributes that can be specified on the queue definition commands.

Cluster attributes available on queue manager definition commands
Cluster attributes that can be specified on queue manager 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.

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 (reset 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.

While a queue manager is suspended from a cluster, it does not receive messages on cluster queues that it hosts if there is an available queue of the same name on an alternative queue manager in the cluster. However, messages that are explicitly targeted at this queue manager, or where the target queue is only available on this queue manager, are still directed to this queue manager.

Receiving further inbound messages while the queue manager is suspended can be prevented by stopping the cluster receiver channels for this cluster. To stop the cluster receiver channels for a cluster, use the FORCE mode of the SUSPEND QMGR command.

Related concepts

Workload balancing in clusters
If a cluster contains more than one instance of the same queue, IBM MQ selects a queue manager to route a message to. It uses the cluster workload management algorithm, and a number of cluster workload-specific attributes, to determine the best queue manager to use.

z/OS Asynchronous behavior of CLUSTER commands on z/OS
The command issuer of a cluster command on z/OS receives confirmation a command has been sent, but not that it has completed successfully.

Related tasks

Maintaining a queue manager

Related reference

Cluster attributes available on channel definition commands
Cluster attributes that can be specified on channel definition commands.

Cluster attributes available on queue definition commands
Cluster attributes that can be specified on the queue definition commands.

Cluster attributes available on queue manager definition commands
Cluster attributes that can be specified on queue manager 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

Workload balancing in clusters

If a cluster contains more than one instance of the same queue, IBM MQ selects a queue manager to route a message to. It uses the cluster workload management algorithm, and a number of cluster workload-specific attributes, to determine the best queue manager to use.

Suitable destinations are chosen, by the cluster workload management algorithm, based on the availability of the queue manager and queue, and on a number of cluster workload-specific attributes associated with channels, queues and queue managers. These attributes are described in the subtopics.

After you configure the cluster workload-specific attributes, if the configuration does not behave as you expected, explore the details of how the algorithm chooses a queue manager. See “The cluster workload management algorithm” on page 142. If the results of this algorithm do not meet your needs, you can

write a cluster workload user exit program, and use this exit to route messages to the queue of your choice in the cluster. See Writing and compiling cluster workload exits.

Table 59. Summary of cluster workload-specific attributes

Attribute name	Description
Channel attributes	
CLWLPRTY	Specifies the priority order for channels for cluster workload distribution.
CLWLRANK	Specifies the rank of channels for cluster workload distribution.
CLWLWGHT	Specifies the weight applied to CLUSSDR and CLUSRCVR channels for cluster workload distribution.
NETPRTY	Specifies the priority for a CLUSRCVR channel.
Queue attributes	
CLWLPRTY	Specifies the priority of local, remote, or alias queues for cluster workload distribution.
CLWLRANK	Specifies the rank of a local, remote, or alias queue for cluster workload distribution.
CLWLUSEQ	Specifies whether a local instance of a queue is given preference as a destination over other instances in a cluster.
Queue manager attributes	
CLWLMRUC	Sets the number of most recently chosen channels.
CLWLUSEQ	Specifies whether a local instance of a queue is given preference as a destination over other instances of the queue in a cluster.

Related concepts

z/OS Asynchronous behavior of CLUSTER commands on z/OS
The command issuer of a cluster command on z/OS receives confirmation a command has been sent, but not that it has completed successfully.

Related reference

Cluster attributes available on channel definition commands
Cluster attributes that can be specified on channel definition commands.

Cluster attributes available on queue definition commands
Cluster attributes that can be specified on the queue definition commands.

Cluster attributes available on queue manager definition commands
Cluster attributes that can be specified on queue manager 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 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.

Cluster workload balancing - channel attributes

An alphabetical list of the channel attributes used in cluster workload balancing.

Note: Specify the cluster workload channel attributes on the cluster-receiver channels at the target queue managers. Any balancing you specify on the matching cluster-sender channels is likely to be ignored. See Cluster channels.

CLWLPRTY (Cluster workload priority)

The CLWLPRTY channel attribute specifies the priority order for channels for cluster workload distribution. The value must be in the range 0-9, where 0 is the lowest priority and 9 is the highest.

Use the CLWLPRTY channel attribute to set a priority order for the available cluster destinations. IBM MQ selects the destinations with the highest priority before selecting destinations with the lowest cluster destination priority. If there are multiple destinations with the same priority, it selects the least recently used destination.

If there are two possible destinations, you can use this attribute to allow failover. Messages go to the queue manager with the highest priority channel. If it becomes unavailable then messages go to the next highest priority queue manager. Lower priority queue managers act as reserves.

IBM MQ checks channel status before prioritizing the channels. Only available queue managers are candidates for selection.

Notes:

- Specify this attribute on the cluster-receiver channel at the target queue manager. Any balancing you specify on the matching cluster-sender channel is likely to be ignored. See Cluster channels.
- The availability of a remote queue manager is based on the status of the channel to that queue manager. When channels start, their state changes several times, with some of the states being less preferential to the cluster workload management algorithm. In practice this means that lower priority (backup) destinations can be chosen while the channels to higher priority (primary) destinations are starting.
- If you need to ensure that no messages go to a backup destination, do not use CLWLPRTY. Consider using separate queues, or CLWLRANK with a manual switch over from the primary to back up.

CLWLRANK (Cluster workload rank)

The **CLWLRANK** channel attribute specifies the rank of channels for cluster workload distribution. The value must be in the range 0-9, where 0 is the lowest rank and 9 is the highest.

Use the **CLWLRANK** channel attribute if you want control over the final destination for messages sent to a queue manager in another cluster. Control the choice of final destination by setting the rank of the channels connecting a queue manager to the gateway queue managers at the intersection of the clusters.

When you set **CLWLRANK**, messages take a specified route through the interconnected clusters towards a higher ranked destination. For example, messages arrive at a gateway queue manager that can send them to either of two queue managers using channels ranked 1 and 2. They are automatically sent to the queue manager connected by a channel with the highest rank, in this case the channel to the queue manager ranked 2.

IBM MQ gets the rank of channels before checking channel status. Getting the rank before checking channel status means that even non-accessible channels are available for selection. It allows messages to be routed through the network even if the final destination is unavailable.

Notes:

- Specify this attribute on the cluster-receiver channel at the target queue manager. Any balancing you specify on the matching cluster-sender channel is likely to be ignored. See Cluster channels.
- If you also used the priority attribute **CLWLPRTY**, IBM MQ selects between available destinations. If a channel is not available to the destination with the highest rank, the message is held on the transmission queue. It is released when the channel becomes available. The message does not get sent to the next available destination in the rank order.

CLWLWGHT (Cluster workload weight)

The CLWLWGHT channel attribute specifies the weight applied to CLUSSDR and CLUSRCVR channels for cluster workload distribution. The value must be in the range 1-99, where 1 is the lowest weight and 99 is the highest.

Use CLWLWGHT to send servers with more processing power more messages. The higher the channel weight, the more messages are sent over that channel.

Notes:

- Specify this attribute on the cluster-receiver channel at the target queue manager. Any balancing you specify on the matching cluster-sender channel is likely to be ignored. See Cluster channels.
- When CLWLWGHT is modified from the default of 50 on any channel, workload balancing becomes dependent on the total number of times each channel was chosen for a message sent to any clustered queue. For more information, see "The cluster workload management algorithm" on page 142.

NETPRTY (Network-connection priority)

The NETPRTY channel attribute specifies the priority for a CLUSRCVR channel. The value must be in the range 0-9, where 0 is the lowest priority and 9 is the highest.

Use the NETPRTY attribute to make one network the primary network, and another network the backup network. Given a set of equally ranked channels, clustering chooses the path with the highest priority when multiple paths are available.

A typical example of using the NETPRTY channel attribute is to differentiate between networks that have different costs or speeds and connect the same destinations.

Note: Specify this attribute on the cluster-receiver channel at the target queue manager. Any balancing you specify on the matching cluster-sender channel is likely to be ignored. See Cluster channels.

Related concepts

The cluster workload management algorithm
The workload management algorithm uses workload balancing attributes and many rules to select the final destination for messages being put onto cluster queues.

Related reference

Cluster workload balancing - queue attributes
An alphabetical list of queue attributes used in cluster workload balancing.

Cluster workload balancing - queue manager attributes
An alphabetical list of queue manager attributes used in cluster workload balancing.

"Cluster attributes available on channel definition commands" on page 125

Cluster attributes that can be specified on channel definition commands.

Cluster workload balancing - queue attributes

An alphabetical list of queue attributes used in cluster workload balancing.

CLWLPRTY

The **CLWLPRTY** queue attribute specifies the priority of local, remote, or alias queues for cluster workload distribution. The value must be in the range 0-9, where 0 is the lowest priority and 9 is the highest.

Use the **CLWLPRTY** queue attribute to set a preference for destination queues. IBM MQ selects the destinations with the highest priority before selecting destinations with the lowest cluster destination priority. If there are multiple destinations with the same priority, it selects the least recently used destination.

IBM MQ obtains the priority of queue managers after checking channel status. Only available queue managers are candidates for selection.

Note:

The availability of a remote queue manager is based on the status of the channel to that queue manager. When channels start, their state changes several times, with some of the states being less preferential to the cluster workload management algorithm. In practice this means that lower priority (backup) destinations can be chosen while the channels to higher priority (primary) destinations are starting.

If you need to ensure that no messages go to a backup destination, do not use **CLWLPRTY**. Consider using separate queues, or **CLWLRANK** with a manual switch over from the primary to back up.

If there are two possible destinations, you can use this attribute to allow failover. The highest priority queue manager receives requests, lower priority queue managers act as reserves. If the highest priority queue manager fails, then the next highest priority queue manager that is available, takes over.

CLWLRANK

The **CLWLRANK** queue attribute specifies the rank of a local, remote, or alias queue for cluster workload distribution. The value must be in the range 0-9, where 0 is the lowest rank and 9 is the highest.

Use the **CLWLRANK** queue attribute if you want control over the final destination for messages sent to a queue manager in another cluster. When you set **CLWLRANK**, messages take a specified route through the interconnected clusters towards a higher ranked destination.

For example, you might have defined two identically configured gateway queue managers to improve the availability of a gateway. Suppose you have defined cluster alias queues at the gateways for a local queue defined in the cluster. If the local queue becomes unavailable, you intend the message to be held at one of the gateways pending the queue becoming available again. To hold the queue at a gateway, you must define the local queue with a higher rank than the cluster alias queues at the gateway.

If you define the local queue with the same rank as the queue aliases and the local queue is unavailable, the message travels between the gateways. On finding the local queue unavailable the first gateway queue manager routes the message to the other gateway. The other gateway tries to deliver the message to the target local queue again. If the local queue is still unavailable, it routes the message back to the first gateway. The message keeps being moved back and forth between the gateways until the target local queue became available again. By giving the local queue a higher rank, even if the queue is unavailable, the message is not rerouted to a destination of lower rank.

IBM MQ obtains the rank of queues before checking channel status. Obtaining the rank before checking channel status means that even non-accessible queues are available for selection. It allows messages to be routed through the network even if the final destination is unavailable.

If you used the priority attribute IBM MQ selects between available destinations. If a channel is not available to the destination with the highest rank, the message is held on the transmission queue. It

is released when the channel becomes available. The message does not get sent to the next available destination in the rank order.

CLWLUSEQ

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

The **CLWLUSEQ** queue attribute is valid only for local queues. It only applies if the message is put by an application, or a channel that is not a cluster channel.

LOCAL
The local queue is the only target of MQPUT, providing the local queue is put enabled. MQPUT behavior depends upon the cluster workload management.

QMGR
The behavior is as specified by the **CLWLUSEQ** queue manager attribute.

ANY
MQPUT treats the local queue the same as any other instance of the queue in the cluster for workload distribution.

Related concepts

The cluster workload management algorithm
The workload management algorithm uses workload balancing attributes and many rules to select the final destination for messages being put onto cluster queues.

Related reference

Cluster workload balancing - channel attributes
An alphabetical list of the channel attributes used in cluster workload balancing.

Cluster workload balancing - queue manager attributes
An alphabetical list of queue manager attributes used in cluster workload balancing.

“Cluster attributes available on queue definition commands” on page 127
Cluster attributes that can be specified on the queue definition commands.

Cluster workload balancing - queue manager attributes

An alphabetical list of queue manager attributes used in cluster workload balancing.

CLWLMRUC

The **CLWLMRUC** queue manager attribute sets the number of most recently chosen channels. The cluster workload management algorithm uses **CLWLMRUC** to restrict the number of active outbound cluster channels. The value must be in the range 1 - 999 999 999.

The initial default value is 999 999 999.

CLWLUSEQ

The **CLWLUSEQ** queue manager attribute specifies whether a local instance of a queue is given preference as a destination over other instances of the queue in a cluster. The attribute applies if the **CLWLUSEQ** queue attribute is set to QMGR.

The **CLWLUSEQ** queue attribute is valid only for local queues. It only applies if the message is put by an application, or a channel that is not a cluster channel.

LOCAL
The local queue is the only target of MQPUT. LOCAL is the default.

ANY
MQPUT treats the local queue the same as any other instance of the queue in the cluster for workload distribution.

Related concepts

The cluster workload management algorithm
The workload management algorithm uses workload balancing attributes and many rules to select the final destination for messages being put onto cluster queues.

Related reference

Cluster workload balancing - channel attributes
An alphabetical list of the channel attributes used in cluster workload balancing.

Cluster workload balancing - queue attributes
An alphabetical list of queue attributes used in cluster workload balancing.

"Cluster attributes available on queue manager definition commands" on page 130
Cluster attributes that can be specified on queue manager definition commands.

The cluster workload management algorithm

The workload management algorithm uses workload balancing attributes and many rules to select the final destination for messages being put onto cluster queues.

The workload management algorithm is exercised every time a choice of destination is required:

- It is used at the point a cluster queue is opened, by using the `MQOO_BIND_ON_OPEN` option.
- It is used each time a message is put to a cluster queue when it is opened with `MQOO_BIND_NOT_FIXED`.
- It is used each time a new message group is started when `MQOO_BIND_ON_GROUP` is used to open a cluster queue.
- For topic host routing, it is used each time a message is published to a clustered topic. If the local queue manager is not a host for this topic, the algorithm is used to choose a host queue manager to route the message through.

The following section describes the workload management algorithm used when determining the final destination for messages being put onto cluster queues. These rules are influenced by the settings applied to the following attributes for queues, queue managers, and channels:

Table 60. Attributes for cluster workload management

Queues	Queue managers	Channels
• CLWLPRTY [1] • CLWLRANK [1] • CLWLUSEQ [1] • PUT / PUB	• CLWLMRUC • CLWLUSEQ [1]	• 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 (that is, channels) to a queue manager that hosts a queue with the name `ObjectName` and is in one of the clusters that the queue manager is a member of.

The algorithm steps through the following rules to eliminate destinations from the list of possible destinations.

1. Remote instances of queues or topics or remote CLUSRCVR channels that do not share a cluster with the local queue manager are eliminated.
2. If a queue or topic name is specified, remote CLUSRCVR channels that are not in the same cluster as the queue or topic are eliminated.

[1] This attribute applies only when choosing a clustered queue, not when choosing a topic.

Note: All remaining queues, topics and channels at this stage are made available to the cluster workload exit, if it is configured.

3. All channels to queue managers or queue manager aliases that have a CLWLRANK less than the maximum rank of all remaining channels or queue manager aliases are eliminated.
4. All queues (not queue manager aliases) with a CLWLRANK less than the maximum rank of all remaining queues are eliminated.
5. If more than one instance of a queue, topic, or queue manager alias remains, and if any are pub put enabled, all those that are put disabled are eliminated.

   Note: If only put disabled instances remain then only inquire operations will succeed, all other operations will fail with MQRC_CLUSTER_PUT_INHIBITED.
6. When choosing a queue, if the resulting set of queues contains the local instance of the queue, the local instance is typically used. The local instance of the queue is used if one of the following conditions are true:
   - The use-queue attribute of the queue, CLWLUSEQ, is set to LOCAL.
   - Both the following statements are true:
     - The use-queue attribute of the queue, CLWLUSEQ, is set to QMGR.
     - The use-queue attribute of the queue manager, CLWLUSEQ, is set to LOCAL.
   - The message is received over a cluster channel rather than by being put by a local application.
   - For locally defined queues that are defined with CLWLUSEQ(ANY), or which inherit that same setting from the queue manager, the following points are true, within the wider set of conditions that apply:
     - The local queue is chosen, based on the `status` of the locally-defined CLUSRCVR channels in the same cluster as the queue. This status is compared to the status of the CLUSSDR channels that would take the message to remotely defined queues of the same name.

       For example, there is one CLUSRCVR in the same cluster as the queue. That CLUSRCVR has STOPPING status, whereas the other queues of the same name in the cluster have RUNNING or INACTIVE status. In this case the remote channels will be chosen, and the local queue is not used.
     - The local queue is chosen based on the `number` of CLUSRCVR channels, in any comparison with CLUSSDR channels of the same status, that would take the message to remotely defined queues of the same name.

       For example, there are four CLUSRCVR channels in the same cluster as the queue, and one CLUSSDR channel. All the channels have the same status of either INACTIVE or RUNNING. Therefore, there are five channels to choose from, and two instances of the queue. Four-fifths (80 percent) of the messages go to the local queue.
7. If more than one queue manager remains, if any are not suspended then all those that are suspended are eliminated.
8. If more than one remote instance of a queue or topic remains, all channels that are inactive or running are included. The state constants are listed:
   - MQCHS_INACTIVE
   - MQCHS_RUNNING
9. If no remote instance of a queue or topic remains, all channels that are in binding, initializing, starting, or stopping state are included. The state constants are listed:
   - MQCHS_BINDING
   - MQCHS_INITIALIZING
   - MQCHS_STARTING
   - MQCHS_STOPPING

10. If no remote instance of a queue or topic remains, all channels that are being tried again are included. The state constant is listed:
    - MQCHS_RETRYING
11. If no remote instance of a queue or topic remains, all channels in requesting, paused, or stopped state are included. The state constants are listed:
    - MQCHS_REQUESTING
    - MQCHS_PAUSED
    - MQCHS_STOPPED
    - MQCHS_SWITCHING
12. If more than one remote instance of a queue or topic on any queue manager remains, channels with the highest NETPRTY value for each queue manager are chosen.
13. All remaining channels and queue manager aliases other than channels and aliases with the highest priority, CLWLPRTY, are eliminated. If any queue manager aliases remain, channels to the queue manager are kept.
14. If a queue is being chosen:
    - All queues other than queues with the highest priority, CLWLPRTY, are eliminated, and channels are kept.
15. The remaining channels are then reduced to no more than the maximum allowed number of most recently-used channels, CLWLMRUC, by eliminating the channels with the lowest values of MQWDR.DestSeqNumber.

    Note: Internal cluster control messages are sent using the same cluster workload algorithm where appropriate.

After the list of valid destinations has been calculated, messages are workload balanced across them, using the following logic:

- When more than one remote instance of a destination remains and all channels to that destination have CLWLWGHT set to the default setting of 50, the least recently used channel is chosen. This approximately equates to a round-robin style of workload balancing when multiple remote instances exist.
- When more than one remote instance of a destination remains and one or more of the channels to those queues has CLWLWGHT set to a non-default setting (even if they all have a matching non-default value), then routing becomes dependent on the relative weightings of each channel and the total number of times each channel has previously been chosen when sending messages.
- When observing the distribution of messages for a single clustered queue with multiple instances, this can appear to lead to an unbalanced distribution across a subset of queue instances. This is because it is the historic use of each cluster sender channel from this queue manager that is being balanced, not just the message traffic for that queue. If this behavior is not desirable, complete one of the following steps:
  - Set CLWLWGHT to 50 on all cluster receiver channels if even distribution is required.
  - Or, if certain queue instances need to be weighted differently from others, define those queues in a dedicated cluster, with defined dedicated cluster receiver channels. This action isolates the workload balancing of these queues from others in the cluster.
- The historic data that is used to balance the channels is reset if any cluster workload attributes of available cluster receiver channels are altered or the status of a cluster receiver channel becomes available. Modification to the workload attributes of manually defined cluster sender channels does not reset the historic data.
- When you are considering cluster workload exit logic, the chosen channel is the one with the lowest MQWDR.DestSeqFactor. Each time a channel is chosen, this value is increased by approximately 1000/CLWLWGHT. If there is more than one channel with the lowest value, one of the channels with the lowest MQWDR.DestSeqNumber value is chosen.

The distribution of user messages is not always exact because administration and maintenance of the cluster causes messages to flow across channels. The result is an uneven distribution of user messages that can take some time to stabilize. Because of the mixture of administration and user messages, place no reliance on the exact distribution of messages during workload balancing.

Related reference

Cluster workload balancing - channel attributes
An alphabetical list of the channel attributes used in cluster workload balancing.

Cluster workload balancing - queue attributes
An alphabetical list of queue attributes used in cluster workload balancing.

Cluster workload balancing - queue manager attributes
An alphabetical list of queue manager attributes used in cluster workload balancing.

Asynchronous behavior of CLUSTER commands on z/OS

The command issuer of a cluster command on z/OS receives confirmation a command has been sent, but not that it has completed successfully.

For both REFRESH CLUSTER and RESET CLUSTER, message CSQM130I is sent to the command issuer indicating that a request has been sent. This message is followed by message CSQ9022I to indicate that the command has completed successfully, in that a request has been sent. It does not indicate that the cluster request has been completed successfully.

Any errors are reported to the z/OS console on the system where the channel initiator is running, they are not sent to the command issuer.

The asynchronous behavior is in contrast to CHANNEL commands. A message indicating that a channel command has been accepted is issued immediately. At some later time, when the command has been completed, a message indicating either normal or abnormal completion is sent to the command issuer.

Related concepts

Workload balancing in clusters
If a cluster contains more than one instance of the same queue, IBM MQ selects a queue manager to route a message to. It uses the cluster workload management algorithm, and a number of cluster workload-specific attributes, to determine the best queue manager to use.

Related tasks

Checking that async commands for distributed networks have finished

Related reference

Cluster attributes available on channel definition commands
Cluster attributes that can be specified on channel definition commands.

Cluster attributes available on queue definition commands
Cluster attributes that can be specified on the queue definition commands.

Cluster attributes available on queue manager definition commands
Cluster attributes that can be specified on queue manager 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 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.

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 61. Channel programs for AIX, Linux, and Windows systems

Program name	Direction of connection	Communication
amqrmppa		Any
runmqlsr	Inbound	Any
amqcrs6a	Inbound	LU 6.2
amqcrsta	Inbound	TCP
runmqchl	Outbound	Any
runmqchi	Outbound	Any

runmqlsr (Run IBM MQ listener), runmqchl (Run IBM MQ channel), and runmqchi (Run IBM MQ channel initiator) are control commands that you can enter at the command line.

amqcrsta is invoked for TCP channels on AIX and Linux systems using inetd, where no listener is started.

amqcrs6a is invoked as a transaction program when using LU6.2

IBM i Intercommunication jobs on IBM i

The following jobs are associated with Intercommunication on IBM i. The names are contained in the following table.

Table 62. Job names and descriptions

Job name	Description
AMQCLMAA	Non-threaded Listener
AMQCRSTA	Non-threaded Responder Job
AMQRMPPA	Channel Pool Job
RUNMQCHI	Channel Initiator
RUNMQCHL	Channel Job
RUNMQLSR	Threaded Listener

IBM i Channel states on IBM i

Channel states are displayed on the Work with Channels panel

Table 63. Channel states on IBM i

State name	Meaning
STARTING	Channel is ready to begin negotiation with target MCA
BINDING	Establishing a session and initial data exchange
REQUESTING	Requester channel initiating a connection

Table 63. Channel states on IBM i (continued)	
State name	**Meaning**
RUNNING	Transferring or ready to transfer
PAUSED	Waiting for message-retry interval
STOPPING	Establishing whether to retry or stop
RETRYING	Waiting until next retry attempt
STOPPED	Channel stopped because of an error or because an end-channel command is issued
INACTIVE	Channel ended processing normally or channel never started
*None	No state (for server-connection channels only)

ALW Example: planning a message channel on AIX, Linux, and Windows

This information provides a detailed example of how to connect two queue managers together so that messages can be sent between them.

About this task

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 AIX and Linux systems use Ctrl-d. Alternatively, use the **end** command.

Figure 7 on page 147 shows the example scenario.

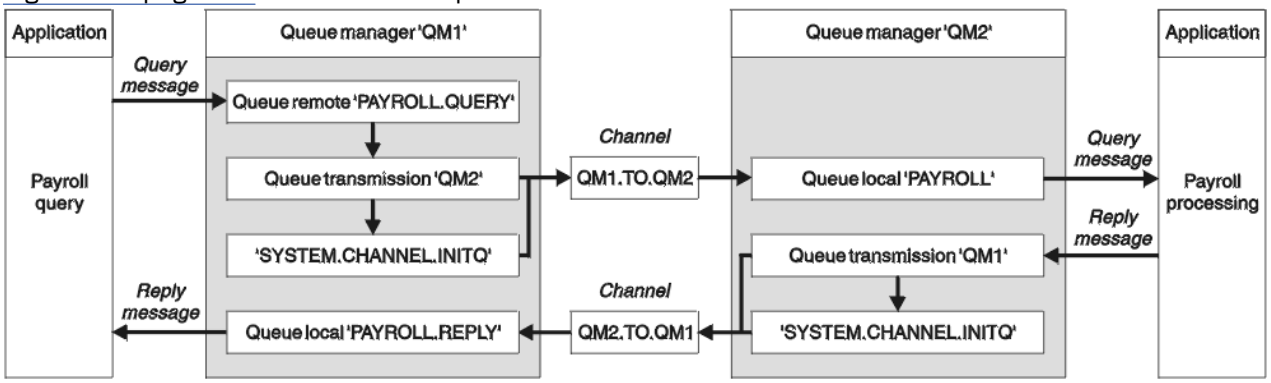

Figure 7. The message channel example for AIX, Linux, and Windows systems

The example involves a payroll query application connected to queue manager QM1 that sends payroll query messages to a payroll processing application running on queue manager QM2. The payroll query application needs the replies to its queries sent back to QM1. The payroll query messages are sent from QM1 to QM2 on a sender-receiver channel called QM1.TO.QM2, and the reply messages are sent back from QM2 to QM1 on another sender-receiver channel called QM2.TO.QM1. Both of these channels are triggered to start as soon as they have a message to send to the other queue manager.

The payroll query application puts a query message to the remote queue "PAYROLL.QUERY" defined on QM1. This remote queue definition resolves to the local queue "PAYROLL" on QM2. In addition, the payroll query application specifies that the reply to the query is sent to the local queue "PAYROLL.REPLY" on QM1. The payroll processing application gets messages from the local queue "PAYROLL" on QM2, and sends the replies to wherever they are required; in this case, local queue "PAYROLL.REPLY" on QM1.

In the example definitions for TCP/IP, QM1 has a host address of 192.0.2.0 and is listening on port 1411, and QM2 has a host address of 192.0.2.1 and is listening on port 1412. The example assumes that these are already defined on your system and available for use.

The object definitions that need to be created on QM1 are:

- Remote queue definition, PAYROLL.QUERY
- Transmission queue definition, QM2 (default=remote queue manager name)
- Sender channel definition, QM1.TO.QM2
- Receiver channel definition, QM2.TO.QM1
- Reply-to queue definition, PAYROLL.REPLY

The object definitions that need to be created on QM2 are:

- Local queue definition, PAYROLL
- Transmission queue definition, QM1 (default=remote queue manager name)
- Sender channel definition, QM2.TO.QM1
- Receiver channel definition, QM1.TO.QM2

The connection details are supplied in the CONNAME attribute of the sender channel definitions.

You can see a diagram of the arrangement in Figure 7 on page 147.

Procedure

See:

- “Setting up the message channel example for AIX, Linux, and Windows” on page 148 for details on setting up the message channels
- “Running and expanding the example for AIX, Linux, and Windows” on page 150 for suggestions on how you can use other products, for example CICS, and how you can connect more applications and user exits.

ALW Setting up the message channel example for AIX, Linux, and Windows

These object definitions allow applications connected to queue manager QM1 to send request messages to a queue called PAYROLL on QM2, and to receive replies on a queue called PAYROLL.REPLY on QM1, and 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.

About this task

All the object definitions have been provided with the DESCR and REPLACE attributes. The other attributes supplied are the minimum required to make the examples work. The attributes that are not supplied take the default values for queue managers QM1 and QM2.

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.

Procedure

- Run the following commands on queue manager QM1:
  a) Setup the 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.

  b) Setup the 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.

  c) Setup the 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)')
```

  d) Setup the receiver channel definition:

```
DEFINE CHANNEL(QM2.TO.QM1) CHLTYPE(RCVR) TRPTYPE(TCP) +
REPLACE DESCR('Receiver channel from QM2')
```

  e) Setup the 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.

- Run the following commands on queue manager QM2:
  a) Setup the 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.

b) Setup the 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.

c) Setup the 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)')
```

d) Setup the receiver channel definition:

```
DEFINE CHANNEL(QM1.TO.QM2) CHLTYPE(RCVR) TRPTYPE(TCP) +
REPLACE DESCR('Receiver channel from QM1')
```

ALW Running and expanding the example for AIX, Linux, and Windows

Information about starting the channel initiator and listener and suggestions for expanding on this scenario.

About this task

Once these definitions have been created, you need to:

- Start the channel initiator on each queue manager.
- Start the listener for each queue manager.

You can also expand the example.

Procedure

1. Start the channel initiator and listener.

   See Setting up communication for Windows and Setting up communication on AIX and Linux systems.
2. You can expand this example by:
   - The use of LU 6.2 communications for interconnection with CICS systems, and transaction processing.
   - Adding more queue, process, and channel definitions to allow other applications to send messages between the two queue managers.
   - Adding user-exit programs on the channels to allow for link encryption, security checking, or additional message processing.
   - Using queue manager aliases and reply-to queue aliases to understand more about how these can be used in the organization of your queue manager network.

IBM i Example: planning a message channel on IBM i

A detailed example of how to connect two IBM i queue managers together so that messages can be sent between them. 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.

About this task

The example illustrates the preparations needed to allow an application using queue manager QM1 to put messages on a queue at queue manager QM2. An application running on QM2 can retrieve these messages, and send responses to a reply queue on QM1.

The example illustrates the use of TCP/IP connections. The example assumes that channels are to be triggered to start when the first message arrives on the transmission queue they are servicing.

This example uses SYSTEM.CHANNEL.INITQ as the initiation queue. This queue is already defined by IBM MQ. You can use a different initiation queue, but you have to define it yourself, start a new instance of the channel initiator using the STRMQMCHLI command, and provide it with the name of your initiation queue. For more information about triggering channels, see Triggering channels.

Note: MQ Adv. CD A message channel that uses TCP/IP can be pointed at an IBM Aspera® faspio Gateway, which provides a fast TCP/IP tunnel that can significantly increase network throughput. See Defining an Aspera gateway connection on Linux or Windows.

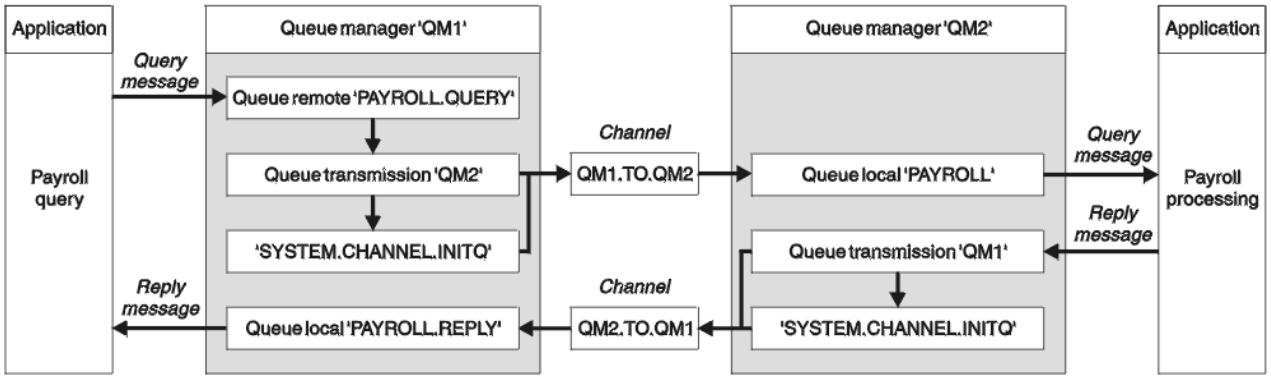

Figure 8. The message channel example for IBM MQ for IBM i

The payroll query messages are sent from QM1 to QM2 on a sender-receiver channel called QM1.TO.QM2, and the reply messages are sent back from QM2 to QM1 on another sender-receiver channel called QM2.TO.QM1. Both of these channels are triggered to start as soon as they have a message to send to the other queue manager.

The payroll query application puts a query message to the remote queue "PAYROLL.QUERY" defined on QM1. This remote queue definition resolves to the local queue "PAYROLL" on QM2. In addition, the payroll query application specifies that the reply to the query is sent to the local queue "PAYROLL.REPLY" on QM1. The payroll processing application gets messages from the local queue "PAYROLL" on QM2, and sends the replies to wherever they are required; in this case, local queue "PAYROLL.REPLY" on QM1.

Both queue managers are assumed to be running on IBM i. In the example definitions, QM1 has a host address of 192.0.2.0 and is listening on port 1411. QM2 has a host address of 192.0.2.1 and is listening on port 1412. The example assumes that these queue managers are already defined on your IBM i system, and are available for use.

The object definitions that need to be created on QM1 are:

- Remote queue definition, PAYROLL.QUERY
- Transmission queue definition, QM2 (default=remote queue manager name)
- Sender channel definition, QM1.TO.QM2
- Receiver channel definition, QM2.TO.QM1
- Reply-to queue definition, PAYROLL.REPLY

The object definitions that need to be created on QM2 are:

- Local queue definition, PAYROLL
- Transmission queue definition, QM1 (default=remote queue manager name)

- Sender channel definition, QM2.TO.QM1
- Receiver channel definition, QM1.TO.QM2

The connection details are supplied in the CONNAME attribute of the sender channel definitions.

You can see a diagram of the arrangement in Figure 8 on page 151.

Procedure

See:

- "Setting up the message channel agent on IBM i" on page 152 for details on setting up the message channels
- "Running and expanding the example for IBM i" on page 155 for suggestions on how you can connect more applications and user exits.

IBM i Setting up the message channel agent on IBM i

The following object definitions allow applications connected to queue manager QM1 to send request messages to a queue called PAYROLL on QM2, to receive replies on a queue called PAYROLL.REPLY on QM1, 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.

About this task

All the object definitions have been provided with the TEXT attributes. The other attributes supplied are the minimum required to make the example work. The attributes that are not supplied take the default values for queue managers QM1 and QM2.

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.

Procedure

- Run the following commands on queue manager QM1:
  a) Setup the remote queue definition by using the CRTMQMQ command with the following attributes:

QNAME	'PAYROLL.QUERY'
QTYPE	*RMT
TEXT	'Remote queue for QM2'
PUTENBL	*YES
TMQNAME	'QM2' (default = remote queue manager name)
RMTQNAME	'PAYROLL'
RMTMQMNAME	'QM2'

  Note: The remote queue definition is not a physical queue, but a means of directing messages to the transmission queue, QM2, so that they can be sent to queue manager QM2.

  b) Setup the transmission queue definition by using the CRTMQMQ command with the following attributes:

QNAME	QM2

QTYPE	*LCL
TEXT	'Transmission queue to QM2'
USAGE	*TMQ
PUTENBL	*YES
GETENBL	*YES
TRGENBL	*YES
TRGTYPE	*FIRST
INITQNAME	SYSTEM.CHANNEL.INITQ
TRIGDATA	QM1.TO.QM2

When the first message is put on this transmission queue, a trigger message is sent to the initiation queue, SYSTEM.CHANNEL.INITQ. The channel initiator gets the message from the initiation queue and starts the channel identified in the named process.

c) Setup the sender channel definition by using the CRTMQCHL command with the following attributes:

CHLNAME	QM1.TO.QM2
CHLTYPE	*SDR
TRPTYPE	*TCP
TEXT	'Sender channel to QM2'
TMQNAME	QM2
CONNAME	'192.0.2.1(1412)'

d) Setup the receiver channel definition by using the CRTMQCHL command with the following attributes:

CHLNAME	QM2.TO.QM1
CHLTYPE	*RCVR
TRPTYPE	*TCP
TEXT	'Receiver channel from QM2'

e) Setup the reply-to queue definition by using the CRTMQMQ command with the following attributes:

QNAME	PAYROLL.REPLY
QTYPE	*LCL
TEXT	'Reply queue for replies to query messages sent to QM2'
PUTENBL	*YES
GETENBL	*YES

The reply-to queue is defined as PUT(ENABLED). This definition ensures that reply messages can be put to the queue. If the replies cannot be put to the reply-to queue, they are sent to the dead-letter queue on QM1 or, if this queue is not available, remain on transmission queue QM1 on queue manager QM2. The queue has been defined as GET(ENABLED) to allow the reply messages to be retrieved.

- Run the following commands on queue manager QM2:

a) Setup the local queue definition by using the CRTMQMQ command with the following attributes:

QNAME	PAYROLL

QTYPE	*LCL
TEXT	'Local queue for QM1 payroll details'
PUTENBL	*YES
GETENBL	*YES

This queue is defined as PUT(ENABLED) and GET(ENABLED) for the same reason as the reply-to queue definition on queue manager QM1.

b) Setup the transmission queue definition by using the CRTMQMQ command with the following attributes:

QNAME	QM1
QTYPE	*LCL
TEXT	'Transmission queue to QM1'
USAGE	*TMQ
PUTENBL	*YES
GETENBL	*YES
TRGENBL	*YES
TRGTYPE	*FIRST
INITQNAME	SYSTEM.CHANNEL.INITQ
TRIGDATA	QM2.TO.QM1

When the first message is put on this transmission queue, a trigger message is sent to the initiation queue, SYSTEM.CHANNEL.INITQ. The channel initiator gets the message from the initiation queue and starts the channel identified in the trigger data.

c) Setup the sender channel definition by using the CRTMQMCHL command with the following attributes:

CHLNAME	QM2.TO.QM1
CHLTYPE	*SDR
TRPTYPE	*TCP
TEXT	'Sender channel to QM1'
TMQNAME	QM1
CONNAME	'192.0.2.0(1411)'

d) Setup the receiver channel definition by using the CRTMQMCHL command with the following attributes:

CHLNAME	QM1.TO.QM2
CHLTYPE	*RCVR
TRPTYPE	*TCP
TEXT	'Receiver channel from QM1'

IBM i Running and expanding the example for IBM i

Information about starting the channel initiator and listener and suggestions for expanding on this scenario.

About this task

Once these definitions have been created, you need to:

- Start the channel initiator on each queue manager.
- Start the listener for each queue manager.

The applications can then send messages to each other. The channels are triggered to start by the first message arriving on each transmission queue, so you do not need to issue the STRMQMCHL command.

You can also expand the example.

Procedure

1. Start the channel initiator and listener.

   See Monitoring and controlling channels on IBM i for details about starting a channel initiator and a listener.
2. You can expand this example by:
   - Adding more queue and channel definitions to allow other applications to send messages between the two queue managers.
   - Adding user exit programs on the channels to allow for link encryption, security checking, or additional message processing.
   - Using queue manager aliases and reply-to queue aliases to understand more about how these objects can be used in the organization of your queue manager network.

   For a version of this example that uses MQSC commands, see “Example: planning a message channel on z/OS” on page 155.

z/OS Example: planning a message channel on z/OS

How to connect z/OS or MVS queue managers together so that messages can be sent between them. This example involves a payroll query application connected to queue manager QM1 that sends payroll query messages to a payroll processing application running on queue manager QM2. The payroll query application needs the replies to its queries sent back to QM1.

About this task

The example illustrates the preparations needed to allow an application using queue manager QM1 to put messages on a queue at queue manager QM2. An application running on QM2 can retrieve these messages, and send responses to a reply queue on QM1.

The example illustrates the use of both TCP/IP and LU 6.2 connections. The example assumes that channels are to be triggered to start when the first message arrives on the transmission queue they are servicing.

Note: MQ Adv. CD A message channel that uses TCP/IP can be pointed at an IBM Aspera faspio Gateway, which provides a fast TCP/IP tunnel that can significantly increase network throughput. See Defining an Aspera gateway connection on Linux or Windows.

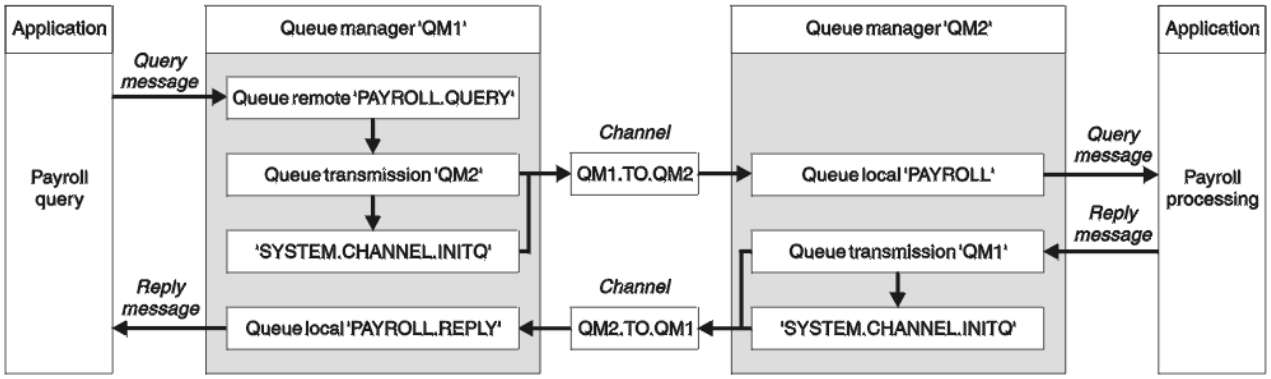

Figure 9. The first example for IBM MQ for z/OS

The payroll query messages are sent from QM1 to QM2 on a sender-receiver channel called QM1.TO.QM2, and the reply messages are sent back from QM2 to QM1 on another sender-receiver channel called QM2.TO.QM1. Both of these channels are triggered to start as soon as they have a message to send to the other queue manager.

The payroll query application puts a query message to the remote queue "PAYROLL.QUERY" defined on QM1. This remote queue definition resolves to the local queue "PAYROLL" on QM2. In addition, the payroll query application specifies that the reply to the query is sent to the local queue "PAYROLL.REPLY" on QM1. The payroll processing application gets messages from the local queue "PAYROLL" on QM2, and sends the replies to wherever they are required; in this case, local queue "PAYROLL.REPLY" on QM1.

Both queue managers are assumed to be running on z/OS. In the example definitions for TCP/IP, QM1 has a host address of 192.0.2.0 and is listening on port 1411, and QM2 has a host address of 192.0.2.1 and is listening on port 1412. In the definitions for LU 6.2, QM1 is listening on a symbolic luname called LUNAME1 and QM2 is listening on a symbolic luname called LUNAME2. The example assumes that these lunames are already defined on your z/OS system and available for use. To define them, see “Example: setting up IBM MQ cross-platform communication on z/OS” on page 45.

The object definitions that need to be created on QM1 are:

- Remote queue definition, PAYROLL.QUERY
- Transmission queue definition, QM2 (default=remote queue manager name)
- Sender channel definition, QM1.TO.QM2
- Receiver channel definition, QM2.TO.QM1
- Reply-to queue definition, PAYROLL.REPLY

The object definitions that need to be created on QM2 are:

- Local queue definition, PAYROLL
- Transmission queue definition, QM1 (default=remote queue manager name)
- Sender channel definition, QM2.TO.QM1
- Receiver channel definition, QM1.TO.QM2

The example assumes that all the SYSTEM.COMMAND.* and SYSTEM.CHANNEL.* queues required to run DQM have been defined as shown in the supplied sample definitions, **CSQ4INSG** and **CSQ4INSX**.

The connection details are supplied in the CONNAME attribute of the sender channel definitions.

You can see a diagram of the arrangement in Figure 9 on page 156.

Procedure

See:

- “Setting up the message channel agent on z/OS” on page 157 for details on setting up the message channels

- “Running and expanding the example for z/OS” on page 159 for suggestions on how you can connect more applications and user exits.

z/OS Setting up the message channel agent on z/OS

The following object definitions allow applications connected to queue manager QM1 to send request messages to a queue called PAYROLL on QM2 and also allows applications to receive replies on a queue called PAYROLL.REPLY on QM1. The definitions also 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.

About this task

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 managers QM1 and QM2.

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.

Procedure

- Run the following commands on queue manager QM1:
  a) Setup the 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.

  b) Setup the transmission queue definition:

```
DEFINE QLOCAL(QM2) DESCR('Transmission queue to QM2') REPLACE +
USAGE(XMITQ) PUT(ENABLED) GET(ENABLED) TRIGGER TRIGTYPE(FIRST) +
TRIGDATA(QM1.TO.QM2) INITQ(SYSTEM.CHANNEL.INITQ)
```

  When the first message is put on this transmission queue, a trigger message is sent to the initiation queue, SYSTEM.CHANNEL.INITQ. The channel initiator gets the message from the initiation queue and starts the channel identified in the trigger data. The channel initiator can only get trigger messages from the SYSTEM.CHANNEL.INITQ queue, so do not use any other queue as the initiation queue.

  c) Setup the sender channel definition:

  For a TCP/IP connection:

```
DEFINE CHANNEL(QM1.TO.QM2) CHLTYPE(SDR) TRPTYPE(TCP) +
REPLACE DESCR('Sender channel to QM2') XMITQ(QM2) +
CONNAME('192.0.2.1(1412)')
```

  For an LU 6.2 connection:

```
DEFINE CHANNEL(QM1.TO.QM2) CHLTYPE(SDR) TRPTYPE(LU62) +
REPLACE DESCR('Sender channel to QM2') XMITQ(QM2) +
CONNAME('LUNAME2')
```

d) Setup the receiver channel definition:

For a TCP/IP connection:

```
DEFINE CHANNEL(QM2.TO.QM1) CHLTYPE(RCVR) TRPTYPE(TCP) +
REPLACE DESCR('Receiver channel from QM2')
```

For an LU 6.2 connection:

```
DEFINE CHANNEL(QM2.TO.QM1) CHLTYPE(RCVR) TRPTYPE(LU62) +
REPLACE DESCR('Receiver channel from QM2')
```

e) Setup the reply-to queue definition:

```
DEFINE QLOCAL(PAYROLL.REPLY) REPLACE PUT(ENABLED) GET(ENABLED) +
DESCR('Reply queue for replies to query messages sent to QM2')
```

The reply-to queue is defined as PUT(ENABLED) which ensures that reply messages can be put to the queue. If the replies cannot be put to the reply-to queue, they are sent to the dead-letter queue on QM1 or, if this queue is not available, remain on transmission queue QM1 on queue manager QM2. The queue has been defined as GET(ENABLED) to allow the reply messages to be retrieved.

- Run the following commands on queue manager QM2:

a) Setup the 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.

b) Setup the transmission queue definition:

```
DEFINE QLOCAL(QM1) DESCR('Transmission queue to QM1') REPLACE +
USAGE(XMITQ) PUT(ENABLED) GET(ENABLED) TRIGGER TRIGTYPE(FIRST) +
TRIGDATA(QM2.TO.QM1) INITQ(SYSTEM.CHANNEL.INITQ)
```

When the first message is put on this transmission queue, a trigger message is sent to the initiation queue, SYSTEM.CHANNEL.INITQ. The channel initiator gets the message from the initiation queue and starts the channel identified in the trigger data. The channel initiator can only get trigger messages from SYSTEM.CHANNEL.INITQ so do not use any other queue as the initiation queue.

c) Setup the sender channel definition:

For a TCP/IP connection:

```
DEFINE CHANNEL(QM2.TO.QM1) CHLTYPE(SDR) TRPTYPE(TCP) +
REPLACE DESCR('Sender channel to QM1') XMITQ(QM1) +
CONNAME('192.0.2.0(1411)')
```

For an LU 6.2 connection:

```
DEFINE CHANNEL(QM2.TO.QM1) CHLTYPE(SDR) TRPTYPE(LU62) +
REPLACE DESCR('Sender channel to QM1') XMITQ(QM1) +
CONNAME('LUNAME1')
```

d) Setup the receiver channel definition:

For a TCP/IP connection:

```
DEFINE CHANNEL(QM1.TO.QM2) CHLTYPE(RCVR) TRPTYPE(TCP) +
REPLACE DESCR('Receiver channel from QM1')
```

For an LU 6.2 connection:

```
DEFINE CHANNEL(QM1.TO.QM2) CHLTYPE(RCVR) TRPTYPE(LU62) +
REPLACE DESCR('Receiver channel from QM1')
```

z/OS Running and expanding the example for z/OS

Information about starting the channel initiator and listener and suggestions for expanding on this example.

About this task

Once these definitions have been created, you need to:

- Start the channel initiator on each queue manager.
- Start the listener for each queue manager.

The applications can then send messages to each other. Because the channels are triggered to start by the arrival of the first message on each transmission queue, you do not need to issue the START CHANNEL MQSC command.

You can also expand the example.

Procedure

1. Start the channel initiator and listener.

   See Starting a channel initiator, and Starting a channel listener for details on how to start a channel initiator and listener.
2. You can expand this example by:
   - Adding more queues, and channel definitions to allow other applications to send messages between the two queue managers.
   - Adding user exit programs on the channels to allow for link encryption, security checking, or additional message processing.
   - Using queue manager aliases and reply-to queue aliases to understand more about how these aliases can be used in the organization of your queue manager network.

z/OS Example: planning a message channel for z/OS using queue sharing groups

This example illustrates the preparations needed to allow an application using queue manager QM3 to put a message on a queue in a queue sharing group that has queue members QM4 and QM5, and also shows the IBM MQ commands (MQSC) that you can use in IBM MQ for z/OS for distributed queuing with queue sharing groups.

About this task

Ensure you are familiar with the example in “Example: planning a message channel on z/OS” on page 155 before trying this one. This example expands the payroll query scenario of that example, to show how to add higher availability of query processing by adding more serving applications to serve a shared queue.

The payroll query application is now connected to queue manager QM3 and puts a query to the remote queue 'PAYROLL QUERY' defined on QM3. This remote queue definition resolves to the shared queue

'PAYROLL' hosted by the queue managers in the queue sharing group QSG1. The payroll processing application now has two instances running, one connected to QM4 and one connected to QM5.

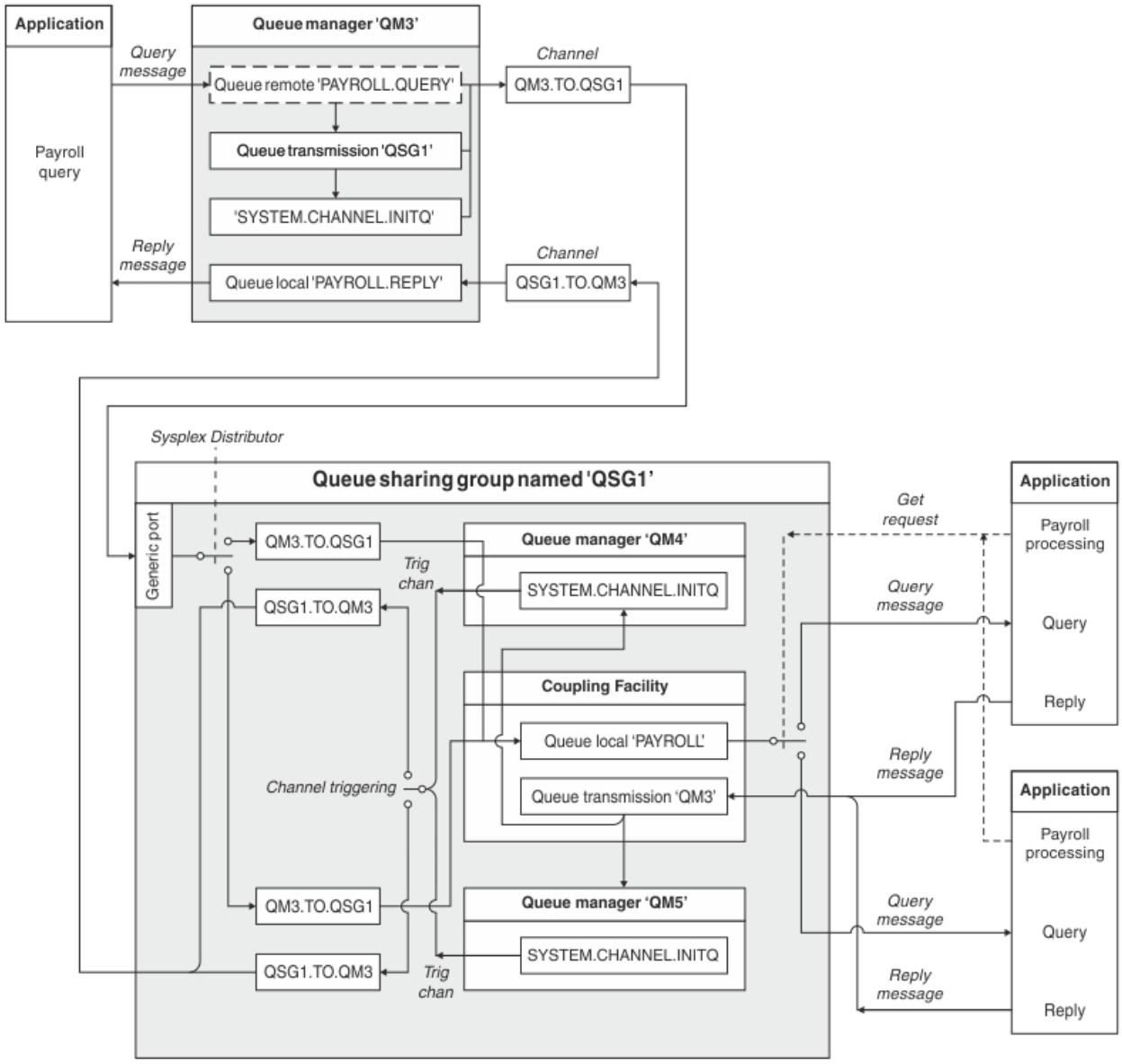

Figure 10. Message channel planning example for IBM MQ for z/OS using queue sharing groups

All three queue managers are assumed to be running on z/OS. In the example definitions for TCP/IP, QM4 has a VIPA address of MVSIP01 and QM5 has a VIPA address of MVSIP02. Both queue managers are listening on port 1414. The generic address that Sysplex Distributor provides for this group is QSG1.MVSIP. QM3 has a host address of 192.0.2.0 and is listening on port 1411.

In the example definitions for LU6.2, QM3 is listening on a symbolic luname called LUNAME1. The name of the generic resource defined for VTAM for the lunames listened on by QM4 and QM5 is LUQSG1. The example assumes that they are already defined on your z/OS system and are available for use. To define them see "Establishing an LU 6.2 connection into a queue sharing group" on page 50.

In this example QSG1 is the name of a queue sharing group, and queue managers QM4 and QM5 are the names of members of the group.

Procedure

See:

- “Setting up the queue sharing group definitions and a queue manager QM3 not in the queue sharing group” on page 161 for details on setting up the definitions.
- “Running the queue sharing group example for z/OS” on page 162 for details on starting the channel initiators and listeners for each queue manager.

z/OS Setting up the queue sharing group definitions and a queue manager QM3 not in the queue sharing group

Producing the following object definitions for one member of the queue sharing group makes them available to all the other members. QM3 is not a member of the queue sharing group.

About this task

Queue managers QM4 and QM5 are members of the queue sharing group. The definitions produced for QM4 are also available for QM5.

The coupling facility list structure is assumed to be called 'APPLICATION1'. If it is not called 'APPLICATION1', you must use your own coupling facility list structure name for the example.

As QM3 is not a member of the queue sharing group you need the object definitions for that queue manager to allow it to put messages to a queue in the queue sharing group.

Procedure

- Setup the shared objects for the queue sharing group definition:
  a) Use the following commands to setup the shared object definitions that are stored in Db2, and their associated messages that are stored within the coupling facility.

```
DEFINE QLOCAL(PAYROLL) QSGDISP(SHARED) REPLACE PUT(ENABLED) GET(ENABLED) +
CFSTRUCT(APPLICATION1) +
DESCR('Shared queue for payroll details')

DEFINE QLOCAL(QM3) QSGDISP(SHARED) REPLACE USAGE(XMITQ) PUT(ENABLED) +
CFSTRUCT(APPLICATION1) +
DESCR('Transmission queue to QM3') TRIGGER TRIGTYPE(FIRST) +
TRIGDATA(QSG1.TO.QM3) GET(ENABLED) INITQ(SYSTEM.CHANNEL.INITQ)
```

- Use the following commands to setup the group object definitions that are stored in Db2®.
  Each queue manager in the queue sharing group creates a local copy of the defined object.
  a) Setup the sender channel:

  Sender channel definition for a TCP/IP connection:

```
DEFINE CHANNEL(QSG1.TO.QM3) CHLTYPE(SDR) QSGDISP(GROUP) TRPTYPE(TCP) +
REPLACE DESCR('Sender channel to QM3') XMITQ(QM3) +
CONNAME('192.0.2.0(1411)')
```

  Sender channel definition for an LU 6.2 connection:

```
DEFINE CHANNEL(QSG1.TO.QM3) CHLTYPE(SDR) QSGDISP(GROUP) TRPTYPE(LU62) +
REPLACE DESCR('Sender channel to QM3') XMITQ(QM3) +
CONNAME('LUNAME1')
```

  b) Setup the receiver channel:

  Receiver channel definition for a TCP/IP connection:

```
DEFINE CHANNEL(QM3.TO.QSG1) CHLTYPE(RCVR) TRPTYPE(TCP) +
REPLACE DESCR('Receiver channel from QM3') QSGDISP(GROUP)
```

  Receiver channel definition for an LU 6.2 connection:

```
DEFINE CHANNEL(QM3.TO.QSG1) CHLTYPE(RCVR) TRPTYPE(LU62) +
REPLACE DESCR('Receiver channel from QM3') QSGDISP(GROUP)
```

- Setup queue manager QM3 object definitions.

  a) Setup the CONNAME

  The CONNAME for this channel is the generic address of the queue sharing group, which varies according to transport type.

  For a TCP/IP connection:

```
DEFINE CHANNEL(QM3.TO.QSG1) CHLTYPE(SDR) TRPTYPE(TCP) +
REPLACE DESCR('Sender channel to QSG1') XMITQ(QSG1) +
CONNAME('QSG1.MVSIP(1414)')
```

  For an LU 6.2 connection:

```
DEFINE CHANNEL(QM3.TO.QSG1) CHLTYPE(SDR) TRPTYPE(LU62) +
REPLACE DESCR('Sender channel to QSG1') XMITQ(QSG1) +
CONNAME('LUQSG1') TPNAME('MQSERIES') MODENAME('#INTER')
```

  b) Setup the other definitions.

  These definitions are required for the same purposes as those used in the sub topics for "Example: planning a message channel on z/OS" on page 155.

```
DEFINE QREMOTE(PAYROLL.QUERY) DESCR('Remote queue for QSG1') REPLACE +
PUT(ENABLED) XMITQ(QSG1) RNAME(APPL) RQMNAME(QSG1)

DEFINE QLOCAL(QSG1) DESCR('Transmission queue to QSG1') REPLACE +
USAGE(XMITQ) PUT(ENABLED) GET(ENABLED) TRIGGER TRIGTYPE(FIRST) +
TRIGDATA(QM3.TO.QSG1) INITQ(SYSTEM.CHANNEL.INITQ)

DEFINE CHANNEL(QSG1.TO.QM3) CHLTYPE(RCVR) TRPTYPE(TCP) +
REPLACE DESCR('Receiver channel from QSG1')

DEFINE CHANNEL(QSG1.TO.QM3) CHLTYPE(RCVR) TRPTYPE(LU62) +
REPLACE DESCR('Receiver channel from QSG1')

DEFINE QLOCAL(PAYROLL.REPLY) REPLACE PUT(ENABLED) GET(ENABLED) +
DESCR('Reply queue for replies to query messages sent to QSG1')
```

z/OS Running the queue sharing group example for z/OS

Information about starting the channel initiators and listeners.

About this task

After you have created the required objects, you need to:

- Start the channel initiator for all three queue managers.
- Start the listeners for both queue managers.

Procedure

1. Start the channel initiators.

   See Starting a channel initiator for details on how to start a channel initiator.
2. Start the listeners.

   See Starting a channel listener for details on how to start a listener.

   For a TCP/IP connection, each member of the group must have a group listener started that is listening on port 1414.

```
STA LSTR PORT(1414) IPADDR(MVSIP01) INDISP(GROUP)
```

The previous entry starts the listener on QM4, for example.

For an LU6.2 connection, each member of the group must have a group listener started that is listening on a symbolic luname. This luname must correspond to the generic resource LUQSG1.

```
STA LSTR PORT(1411)
```

The previous entry starts the listener on QM3.

Using an alias to refer to an MQ library

You can define an alias to refer to an MQ library in your JCL, rather than use the name of the MQ library directly. Then, if the name of the MQ library changes, you have only to delete and redefine the alias.

Example

The following example defines an alias MQM.SCSQANLE to refer to the MQ library MQM.V600.SCSQANLE:

```
//STEP1 EXEC PGM=IDCAMS
//SYSPRINT DD SYSOUT=*
//SYSIN DD *
DELETE (MQM.SCSQANLE)
DEFINE ALIAS (NAME(MQM.SCSQANLE) RELATE(MQM.V600.SCSQANLE))
/*
```

Then, to refer to the MQM.V600.SCSQANLE library in your JCL, use the alias MQM.SCSQANLE.

Note: The library and alias names must be in the same catalog, so use the same high level qualifier for both; in this example, the high level qualifier is MQM.

Managed File Transfer configuration reference

Reference information to help you configure Managed File Transfer.

The use of environment variables in MFT properties

It is possible for environment variables to be used in Managed File Transfer properties that represent file or directory locations. This allows the locations of files or directories used when running parts of the product to vary depending on the current environment (such as the user running a command, for example).

The following properties accept file or directory locations and can therefore contain environment variables:

- `agentQMgrAuthenticationCredentialsFile`
- `agentSslKeyStore`
- `agentSslKeyStoreCredentialsFile`
- `agentSslTrustStore`
- `agentSslTrustStoreCredentialsFile`
- `cdNodeKeystoreCredentialsFile`
- `cdNodeTruststoreCredentialsFile`
- `cdTmpDir`
- `cdNodeKeystore`
- `cdNodeTruststore`
- `commandPath`

- connectionQMgrAuthenticationCredentialsFile
- connectionSslKeyStore
- connectionSslKeyStoreCredentialsFile
- connectionSslTrustStore
- connectionSslTrustStoreCredentialsFile
- coordinationSslKeyStore
- coordinationSslKeyStoreCredentialsFile
- coordinationQMgrAuthenticationCredentialsFile
- coordinationSslTrustStore
- coordinationSslTrustStoreCredentialsFile
- exitClassPath
- exitNativeLibraryPath
- javaCoreTriggerFile
- loggerQMgrAuthenticationCredentialsFile
- sandboxRoot
- transferRoot
- wmqfte.database.credentials.file

Example for Windows

In this example on a Windows system, a user `fteuser` using an environment variable of USERPROFILE:

```
wmqfte.database.credentials.file=%USERPROFILE%\\logger\\mqmftcredentials.xml
```

Resolves to the following file path:

```
C:\Users\fteuser\logger\mqmftcredentials.xml
```

Example for AIX and Linux

In this example on a UNIX system, a user `fteuser` using an environment variable of HOME:

```
transferRoot=$HOME/fte/
```

Resolves to the following file path:

```
/home/fteuser/fte/
```

Related reference

"The MFT coordination.properties file" on page 191
The `coordination.properties` file specifies the connection details to the coordination queue manager. Because several Managed File Transfer installations might share the same coordination queue manager, you can use a symbolic link to a common `coordination.properties` file on a shared drive.

"The MFT command.properties file" on page 195
The `command.properties` file specifies the command queue manager to connect to when you issue commands and the information that Managed File Transfer requires to contact that queue manager.

"The MFT agent.properties file" on page 169

Each Managed File Transfer Agent has its own properties file, `agent.properties`, that must contain the information that an agent uses to connect to its queue manager. The `agent.properties` file can also contain properties that alter the behavior of the agent.

SSL/TLS properties for MFT

"The MFT logger.properties file" on page 199
The Managed File Transfer logger has a set of configuration properties. Specify these properties in the `logger.properties` file, which is in the *MQ_DATA_PATH*`/mqft/config/`*coordination_qmgr_name*`/loggers/`*logger_name* directory.

MFT Agent properties for user exits

Protocol bridge properties file format

Connect:Direct process definition file format

Connect:Direct node properties file format

The MFT installation.properties file

The `installation.properties` file specifies the name of your default set of configuration options. This entry points Managed File Transfer to a structured set of directories and property files that contain the configuration to use. Typically the name of a set of configuration options is the name of the associated coordination queue manager.

This file is created by the installer, and can be changed by using the **fteChangeDefaultConfigurationOptions** command.

The `installation.properties` file is located in your *MQ_DATA_PATH* directory. For example on Windows, the default file location is *MQ_DATA_PATH*`\mqft\installations\`*installation_name* and on AIX and Linux systems, the default file location is `/var/mqm/mqft/installations/`*installation_name*.

For the Redistributable Managed File Transfer Agent, the data path is set when you run the **fteCreateEnvironment** command. If you run the command and specify your chosen location with the **-d** parameter, the data path is set for this location. If you do not specify the location with the **fteCreateEnvironment** command, a directory `mftdata` is created under the root directory where the Redistributable Managed File Transfer Agent is extracted. The `installation.properties` file for the Redistributable Managed File Transfer Agent is located in the `MQ_DATA_PATH\mqft\installations\MFTZipInstall` directory.

The `installation.properties` file contains the following values:

Table 64. Basic properties

Property name	Description	Default value
commandMessagePriority	Sets the priority of both internal messages and command messages for the **fteStopAgent**, **fteCancelTransfer** and **ftePingAgent** commands. If you submit a large number of transfer requests to transfer many small files in quick succession, for example, the new transfer requests can become queued on the source agent's command queue. The external and internal messages have the default IBM MQ message priority so the internal messages are blocked by the new transfer requests. This can cause the transfer negotiation time to be exceeded and for the transfers to go into recovery. You can also use the commandMessagePriority property to set the priority of internal acknowledgment and acknowledgment-expected messages. To prioritize the internal Managed File Transfer messages above new transfer requests, set this property to a value between 1 (the lowest) and 9 (the highest). The default value of the commandMessagePriority property is 8. This means that, if the IBM MQ attribute DEFPRTY (default priority) on an agent command queue is less than or equal to 7, internal negotiation messages are prioritized ahead of new transfer requests. If the value of the DEFPRTY attribute is set to either 8 or 9, to maintain the effectiveness of the commandMessagePriority property, you must change either DEFPRTY or the commandMessagePriority property.	For IBM MQ 9.0.0.0 and later, the default value is 8. For earlier releases, and before APAR IT06213, the default value is the MQPRI_PRIORITY_AS_Q_DEF constant, which has a value of -1.

Table 64. Basic properties (continued)

Property name	Description	Default value
commonCredentialsKeyFile	The fully qualified path name of the file containing the credential key used while encrypting credentials. The most common name of the MFT credentials file is `MQMFTCredentials.xml`. For more information on using the commonCredentialsKeyFile property, see Decrypting credentials.	The fully-qualified path of the key file
defaultProperties	The name of the default set of configuration options. This value is the name of a directory located in the configuration directory, which contains directories and properties files that specify configuration information.	No default
enableFunctionalFixPack	The fix pack function level to enable. By default, any new function included with a fix pack is not enabled. Set this property to a version identifier to enable the new features available with that version. You can specify the version identifier with or without period characters (.). For example, to use the function available with IBM MQ 8.0.0 Fix Pack 2, set this property to `8002` or `8.0.0.2`.	No default

Table 64. Basic properties (continued)		
Property name	**Description**	**Default value**
messagePublicationFormat	Allows you to specify the message publication format used by MFT agents for their status XML messages. This property can be set to the following values: **messagePublicationFormat=mixed** Messages are published without an MQMD FORMAT (MQFMT_NONE), except for those messages that are published under the /LOG topic tree, which are published in the MQMD format of MQFMT_STRING. **messagePublicationFormat=MQFMT_NONE** Messages are published without an MQMD FORMAT. **messagePublicationFormat=MQFMT_STRING** Messages are published in a string format.	messagePublicationFormat=mixed
z/OS z/OS **z/OS-specific:**		
z/OS **productId**	Product type against which MFT usage is to be recorded: • Standalone Managed File Transfer product. (MFT is the **productID**). • Part of an IBM MQ Advanced product. (ADVANCED is the **productID**). • Part of an IBM MQ Advanced for z/OS Value Unit Edition product. (ADVANCEDVUE is the **productID**). See Reporting product information for more information on product usage recording. Multi This property is ignored on Multiplatforms.	MFT

The following text is an example of the contents of a `installation.properties` file.

```
defaultProperties=ERIS
```

ERIS is the name of a directory that is located in the same directory as the `installation.properties` file. The directory ERIS contains directories and properties files that describe a set of configuration options.

Related concepts

MFT configuration options on Multiplatforms

Related reference

fteChangeDefaultConfigurationOptions

The MFT `agent.properties` file

Each Managed File Transfer Agent has its own properties file, `agent.properties`, that must contain the information that an agent uses to connect to its queue manager. The `agent.properties` file can also contain properties that alter the behavior of the agent.

The `agent.properties` file is created by the installer or by the **`fteCreateAgent`**, **`fteCreateBridgeAgent`** or **`fteCreateCDAgent`** command. You can use any of these commands with the **`-f`** flag to change the basic agent queue manager properties and those advanced agent properties that are associated with the type of agent that you are creating. To change or add advanced agent properties, you must edit the file in a text editor.

On Multiplatforms, the `agent.properties` file for an agent is in the *`MQ_DATA_PATH`*`/mqft/config/`*`coordination_qmgr_name`*`/agents/`*`agent_name`* directory.

On z/OS, the `agent.properties` file location is `$BFG_CONFIG variable/mqft/config/coordination_qmgr_name/agents/agent_name`.

If you change the `agent.properties` file you must restart the agent to pick up the changes.

You can use environment variables in some Managed File Transfer properties that represent file or directory locations. This allows you to use the locations of files or directories when running parts of the product to vary depending on environment changes, such as which user is running the process. For more information, see “The use of environment variables in MFT properties” on page 163.

Note: On Windows, two properties:

- `windowsService`
- `windowsServiceVersion`

are added into the `agent.properties` file by the MFT commands used to set up an agent to run as a Windows service.

You should not add the properties, or modify them, manually as this will prevent the agent from working properly.

Basic agent properties

Each MFT `agent.properties` file contains the following basic agent properties:

Table 65. Basic agent properties

Property name	Description	Default value
agentName	The name of the agent. The name of the agent must conform to the IBM MQ object naming conventions. For more information, see “MFT object naming conventions” on page 225.	No default
agentDesc	The description of the agent - if you choose to create a description.	No default
agentQMgr	The agent queue manager name.	No default
agentQMgrHost	The host name or IP address of the agent queue manager.	No default
agentQMgrPort	The port number that is used for client connections to the agent queue manager.	1414
agentQMgrChannel	The SVRCONN channel name that is used to connect to the agent queue manager.	SYSTEM.DEF.SVRCONN

Table 65. Basic agent properties (continued)		
Property name	**Description**	**Default value**
agentType	The type of agent: • Standard non-bridge agent (STANDARD) • Protocol bridge agent (BRIDGE) • Connect:Direct® bridge agent (CD_BRIDGE) • Embedded agent as used by IBM Integration Bus (EMBEDDED) • Sterling File Gateway embedded agent (SFG)	STANDARD

If you do not specify a value for the agentQMgrHost property, bindings mode is used by default.

If you specify a value for the agentQMgrHost property but do not specify values for the agentQMgrPort and agentQMgrChannel properties, a port number of 1414 and a channel of SYSTEM.DEF.SVRCONN are used by default.

Advanced agent properties

Managed File Transfer also provides more advanced agent properties that help you configure agents. If you want to use any of the following properties, manually edit the `agent.properties` file to add the required advanced properties. Parentheses, commas (,) and backslashes (\) are special characters in MFT commands, and must be escaped with a backslash (\) character. Windows File paths on Windows can be specified either using double backslashes (\\) as a separator, or using single forward slashes (/). For more information about character escaping in Java properties files, see the Oracle documentation Javadoc for the Properties class.

- Agent size properties
- Code page properties
- Command properties
- Connection properties
- Connect:Direct bridge properties
- File to message and message to file agent properties
- General agent properties
- High availability properties
- Input/output properties
- V 9.3.0 Transfer log properties
- Multi-channel support properties
- Multi-instance properties
- Process controller properties
- Protocol bridge properties
- Protocol bridge agent log properties
- Queue properties
- Resource monitoring properties
- Root directory properties
- Scheduler property
- Security properties
- SSL/TLS properties
- Timeout properties
- Transfer recovery timeout properties

- Trace and logging properties
- Transfer limit properties
- User exit routine properties
- IBM MQ client compression properties
- z/OS z/OS-specific properties
- Other properties

Table 66. Advanced agent properties: Agent size

Property name	Description	Default value
agentCheckpointInterval	The interval in complete frames of data between which a checkpoint is taken for recovery purposes. This is an advanced property and for most Managed File Transfer configurations it is not necessary to modify its value. If there is a problem which causes the transfer to go into recovery, the transfer can recover only to a checkpoint boundary. Hence, the larger this value (with large agentChunkSize, agentWindowSize, and agentFrameSize values), the longer the time that is needed for the agent to recover transfers. For reliable Managed File Transfer networks where transfers rarely enter a recovery state, it may be beneficial to increase this value to increase overall performance.	1
agentChunkSize	The size of each transfer chunk for the transport of file data. Hence, denotes the maximum size of the IBM MQ messages that are transferred between the source and the destination agents. This is an advanced property and for most Managed File Transfer configurations it is not necessary to modify its value. This value is negotiated between the source agent and the destination agent, and the larger of the two values is used. If you want to change the value of this property, change the value at both the source agent and at the destination agent. agentChunkSize is an integer value. For example: agentChunkSize = 10240 sets the chunk size to 10 KB.	262144-byte (which is equivalent to 256 KB)
agentFrameSize	The number of windows for the transfer frame. This is an advanced property and for most Managed File Transfer configurations it is not necessary to modify its value. For networks that have high latency, increasing this value may improve overall performance as it causes the agent to have more message chunks active concurrently. The value of this property, multiplied by agentWindowSize, multiplied by agentChunkSize, denotes the upper limit of the memory consumption of the agent for each transfer. For example, 262144-byte chunks x 10 x 5 = 12.5 MB for each transfer. Note: If the size of the files that is transferred in a single transfer is less than 12.5 MB increasing this property has no effect on the performance of the transfer.	5
agentWindowSize	The number of chunks for each window. This is an advanced property and for most Managed File Transfer configurations it is not necessary to modify its value. For networks that have high latency, increasing this value may improve overall performance. This is because it causes the agent to have more message chunks active concurrently and reduces the frequency that acknowledgment messages are sent back to the source agent. The value of this property, multiplied by agentFrameSize, multiplied by agentChunkSize, denotes the upper limit of the memory consumption of the agent for each transfer, and denotes the upper limit of the IBM MQ message data on the data queue of the destination agent. For example, 262144-byte chunks x 10 x 5 = an upper limit of 12.5 MB, for each transfer. Note: If the size of the files that is transferred in a single transfer is less than 12.5 MB increasing the value of this property has no effect on the performance of the transfer.	10

Table 67. Advanced agent properties: Code page

Property name	Description	Default value
agentCcsid	The code page the agent connects to its agent queue manager with. If you specify a value for agentCcsid, you must also specify a value for agentCcsidName. For information on how to view the known code pages for the JVM, see the -hsc parameter in the **fteCreateBridgeAgent** command.	1208

Table 67. Advanced agent properties: Code page (continued)		
Property name	**Description**	**Default value**
agentCcsidName	The Java representation of the agentCcsid. If you specify a value for agentCcsidName, you must also specify a value for agentCcsid.	UTF8

Table 68. Advanced agent properties: Command		
Property name	**Description**	**Default value**
maxCommandHandlerThreads	Controls the number of threads available for the initial parsing and processing of transfer command messages. When active, the threads require a connection to the queue manager but the threads release the connection when idle.	5
maxCommandOutput	The maximum number of bytes stored for command output. This property applies to commands specified for a managed call and preSource, postSource, preDestination, and postDestination commands for a managed transfer. This limits the length of command output that is written to the transfer log on the SYSTEM.FTE topic.	10240
maxCommandRetries	The maximum number of retries for a command that the agent permits. This property applies to commands specified for a managed call and the preSource, postSource, preDestination, and postDestination commands for a managed transfer.	9
maxCommandWait	The maximum wait, in seconds, between retries that the agent permits. This property applies to commands specified for a managed call and the preSource, postSource, preDestination, and postDestination commands for a managed transfer.	60
immediateShutdownTimeout	For an immediate shutdown of an agent, you can use this property to specify the maximum amount of time in seconds an agent waits for its transfers to complete before forcing a shutdown. **Note:** Do not change the value of this property to less than the default of 10 seconds. An immediate shutdown of an agent requires sufficient time to end any external processes. If the value of this property is too low, processes might be left running. If the value 0 is specified for this property, the agent waits for all outstanding transfers to stop. If an invalid value is specified for this property, the default value is used.	10

Table 69. Advanced agent properties: Connection		
Property name	**Description**	**Default value**
javaLibraryPath	When connecting to a queue manager in bindings mode, Managed File Transfer must have access to the IBM MQ Java bindings libraries. By default Managed File Transfer looks for the bindings libraries in the default location that is defined by IBM MQ. If the bindings libraries are in a different location, use this property to specify the location of the bindings libraries.	None

Table 70. Advanced agent properties: Connect:Direct bridge		
Property name	**Description**	**Default value**
cdNode	Required property if you want to use the Connect:Direct bridge. The name of the Connect:Direct node to use to transfer messages from the Connect:Direct bridge agent to destination Connect:Direct nodes. This node is part of the Connect:Direct bridge, not the remote node that is the source or destination of the transfer. For more information, see The Connect:Direct bridge.	No default
cdNodeHost	The host name or IP address of the Connect:Direct node to use to transfer files from the Connect:Direct bridge agent to destination nodes (the Connect:Direct bridge node). In most cases, the Connect:Direct bridge node is on the same system as the Connect:Direct bridge agent. In these cases, the default value of this property, which is the IP address of the local system, is correct. If your system has multiple IP addresses, or your Connect:Direct bridge node is on a different system to your Connect:Direct bridge agent and their systems share a file system, use this property to specify the correct host name for the Connect:Direct bridge node. If you have not set the cdNode property, this property is ignored.	The host name or IP address of the local system

Table 70. Advanced agent properties: Connect:Direct bridge (continued)

Property name	Description	Default value
cdNodePort	The port number of the Connect:Direct bridge node that client applications use to communicate with the node. In Connect:Direct product documentation, this port is referred to as the API port. If you have not set the cdNode property, this property is ignored.	1363
cimpDir	The location to store files temporarily on the system where the Connect:Direct bridge agent is running before they are transferred to the destination Connect:Direct node. This property specifies the full path of the directory where files are temporarily stored. For example, if cdTmpDir is set to `/tmp` then the files are temporarily placed in the `/tmp` directory. The Connect:Direct bridge agent and the Connect:Direct bridge node must be able to access the directory specified by this parameter using the same path name. Consider this when planning the installation of your Connect:Direct bridge. If possible, create the agent on the system where the Connect:Direct node that is part of the Connect:Direct bridge is located. If your agent and node are on separate systems, the directory must be on a shared file system and be accessible from both systems using the same path name. For more information about the supported configurations, see The Connect:Direct bridge. If you have not set the cdNode property, this property is ignored. The value of this property can contain environment variables. See "The use of environment variables in MFT properties" on page 163 for more information.	*value_of_java.io.tmpdir*`/cdbridge-`*agentName*
cdTrace	Whether the agent traces data that is sent between the Connect:Direct bridge agent and its Connect:Direct node. The value of this property can be `true` or `false`.	false
cdMaxConnectionRetries	The maximum number of Connect:Direct connection attempts, for a file transfer where a successful connection has not yet been made, before the transfer fails.	-1 (an infinite number of attempts)
cdMaxPartialWorkConnectionRetries	The maximum number of Connect:Direct connection attempts, for a file transfer where a previous connection attempt has been successful and transfer work has completed, before the transfer fails.	-1 (an infinite number of attempts)
cdMaxWaitForProcessEndStats	The maximum time in milliseconds to wait for Connect:Direct process completion information to become available within the Connect:Direct node statistics information, after the process has ended, before the file transfer is judged to have failed. Typically the information is available immediately, but under certain failure conditions the information is not published. In these conditions the file transfer fails after waiting for the amount of time that is specified by this property.	60000
cdAppName	The application name that the Connect:Direct bridge agent uses to connect to the Connect:Direct node that is part of the bridge.	Managed File Transfer *current version*, where *current version* is the version number of the product.
cdNodeLocalPortRange	The range of local ports to use for socket connections between the Connect:Direct bridge agent and the Connect:Direct node that is part of the bridge. The format of this value is a comma-separated list of values or ranges. By default, the operating system selects the local port numbers.	None
cdNodeProtocol	The protocol that the Connect:Direct bridge agent uses to connect to the Connect:Direct node that is part of the bridge. The following values are valid: • TCPIP • SSL • TLS	TCPIP
cdNodeKeystore	The path to the keystore that is used for secure communications between the Connect:Direct bridge agent and the Connect:Direct node that is part of the bridge. If you have not set the cdNodeProtocol property to SSL or TLS, this property is ignored. The value of this property can contain environment variables. See "The use of environment variables in MFT properties" on page 163 for more information.	None

Table 70. Advanced agent properties: Connect:Direct bridge (continued)		
Property name	**Description**	**Default value**
cdNodeKeystoreType	The file format of the keystore that is specified by the cdNodeKeystore property. The following values are valid: jks and pkcs12. If you have not set the cdNodeProtocol property to SSL or TLS, this property is ignored.	jks
cdNodeKeystoreCredentialsFile	The path to the file that contains the cdNodeKeystore credentials. The value of this property can contain environment variables. See "The use of environment variables in MFT properties" on page 163 for more information. ALW For details on creating credentials files see MFT and IBM MQ connection authentication z/OS For details on creating credentials files see Configuring MQMFTCredentials.xml on z/OS.	The default value for this property is: z/OS Linux AIX `$HOME/MQMFTCredentials.xml` Windows `%USERPROFILE%/MQMFTCredentials.xml`
cdNodeTruststore	The path to the truststore that is used for secure communications between the Connect:Direct bridge agent and the Connect:Direct node that is part of the bridge. If you have not set the cdNodeProtocol property to SSL or TLS, this property is ignored. The value of this property can contain environment variables. See "The use of environment variables in MFT properties" on page 163 for more information.	None
cdNodeTruststoreType	The file format of the truststore that is specified by the cdNodeTruststore property. The following values are valid: jks and pkcs12. If you have not set the cdNodeProtocol property to SSL or TLS, this property is ignored.	jks
cdNodeTruststoreCredentialsFile	The path to the file that contains the cdNodeTruststore credentials. The value of this property can contain environment variables. See "The use of environment variables in MFT properties" on page 163 for more information. ALW For details on creating credentials files see MFT and IBM MQ connection authentication z/OS For details on creating credentials files see Configuring MQMFTCredentials.xml on z/OS.	The default value for this property is: z/OS Linux AIX `$HOME/MQMFTCredentials.xml` Windows `%USERPROFILE%/MQMFTCredentials.xml`
logCDProcess	The level of Connect:Direct process logging that is recorded in the agent event log in the `output0.log` file. The values that this property can have are `None` or `Failures` or `All`.	None

Table 71. Advanced agent properties: File to message and message to file agent		
Property name	**Description**	**Default value**
deleteTmpFileAfterRenameFailure	Setting this property to a value of false ensures that temporary files are not deleted from the destination if the rename operation fails. In this case, the transferred data remains at the destination in a temporary (.part) file. You can manually rename this file later. By default this property has the value of true. This property applies to both message-to-file and file-to-file transfers.	true
enableQueueInputOutput	By default, the agent cannot read data from a source queue or write data to a destination queue as part of a transfer. Setting this value to true enables the agent to perform file to message, and message to file transfers. The value of this property can be `true` or `false`.	false
enableSystemQueueInputOutput	Specifies whether the agent can read from or write to IBM MQ system queues. System queues are prefixed with the qualifier SYSTEM. **Note:** System queues are used by IBM MQ, Managed File Transfer, and other applications to transmit important information. Changing this property enables the agent to access these queues. If you enable this property, use user sandboxing to limit the queues that the agent can access.	false

Table 71. Advanced agent properties: File to message and message to file agent (continued)

Property name	Description	Default value
enableClusterQueueInputOutput	Specifies whether the agent can read from or write to IBM MQ clustered queues. **Note:** You must specify the enableClusterQueueInputOutput agent property in addition to the enableQueueInputOutput property.	false
maxDelimiterMatchLength	The maximum number of characters that can be matched by the Java regular expression that is used to split a text file into multiple messages as part of a file-to-message transfer.	5
maxInputOutputMessageLength	The maximum length, in bytes, of a message that is read from a source queue or written to a destination queue by an agent. The maxInputOutputMessageLength property of the source agent in a transfer determines how many bytes can be read from a message on the source queue. The maxInputOutputMessageLength property of the destination agent in a transfer determines how many bytes can be written to a message on the destination queue. If the length of the message exceeds the value of this property the transfer fails with an error. This property does not affect the Managed File Transfer internal queues. For information about changing this property, see Guidance for setting MQ attributes and MFT properties associated with message size.	1048576
monitorGroupRetryLimit	The maximum number of times that a monitor triggers a message-to-file transfer again if the message group still exists on the queue. The number of times that the message-to-file transfer triggers is determined from the MQMD backout count of the first message in the group. If the agent is restarted the monitor triggers a transfer again even if the number of times the transfer triggers exceeds the value of monitorGroupRetryLimit. If this behavior causes the number of times that the transfer triggers to exceed the value of monitorGroupRetryLimit, the agent writes an error to its event log. If the value -1 is specified for this property, the monitor triggers the transfer again an unlimited number of times until the trigger condition is not satisfied.	10

Table 72. Advanced agent properties: General

Property name	Description	Default value
agentStatusPublishRateLimit	The maximum rate in seconds that the agent republishes its status because of a change in file transfer status. If you set this property to too small a value, the performance of the IBM MQ network might be negatively affected.	30
agentStatusPublishRateMin	The minimum rate in seconds that the agent publishes its status. This value must be greater than or equal to the value of the **agentStatusPublishRateLimit** property.	300
enableMemoryAllocationChecking	The value of this property can be `true` or `false`. It determines whether the Managed File Transfer Agent checks that there is sufficient memory available to run a transfer before a transfer is accepted. The check is made on both the source and destination agents. If there is insufficient memory available, the transfer is rejected. When calculating the memory required for a transfer, the maximum memory that is required by the transfer is used. Therefore, the value might be greater than the actual memory that is used by the transfer. For this reason, the number of concurrent transfers that can run might be reduced if the **enableMemoryAllocationChecking** property is set to true. You are recommended to set the property to true only if you are experiencing problems with Managed File Transfer failing with out-of-memory errors. The transfers that are likely to consume large amounts of memory are file-to-message and message-to-file transfers where the sizes of the messages are large.	false

Table 72. Advanced agent properties: General (continued)

Property name	Description	Default value
enableDetailedReplyMessages	The value of this property can be `true` or `false`. Setting this property to `true` enables managed transfer request replies to contain detailed information about the transferred files. The detailed information and format is the same as that published to the transfer log in the progress messages, that is, the <transferSet> element. For more information, see File transfer log message formats. The detailed reply information is included only when the managed transfer request specifies that detailed reply information is required. To specify this requirement, set the **detailed** attribute of the <reply> element of the managedTransfer XML request message sent to the source agent. For more information, see File transfer request message format. Multiple reply messages can be generated for each transfer request. This number is equal to the number of transfer log progress messages for the transfer plus 1 (where the first reply message is a simple ACK reply). Detailed information is included in all messages, except for the ACK reply messages, but the overall transfer result is included only in the last detailed reply message.	true
enableUserMetadataOptions	The value of this property can be `true` or `false`. It determines whether you can use known keys for user-defined metadata in new transfer requests to provide more transfer options. These known keys always start with the following prefix `com.ibm.wmqfte.`. As a consequence when the **enableUserMetadataOptions** property is set to `true`, keys that use this prefix are not supported for user-defined use. When the **enableUserMetadataOptions** property is set to `true`, the keys that are supported currently are as follows: **com.ibm.wmqfte.insertRecordLineSeparator** For text transfers. When this key is set to `true`, specifies that when reading record-oriented files, such as z/OS data sets, line separators are to be inserted between records. When this key is set to `false`, specifies that when reading record-oriented files, line separators are not to be inserted between records. **com.ibm.wmqfte.newRecordOnLineSeparator** For text transfers. When this key is set to `true`, specifies that when writing to record-oriented files, such as z/OS data sets, that line separators indicate a new record and are not written as part of the data. When this key is set to `false`, specifies that, when writing to record-oriented files, line separators are to be treated like any other character (that is, no record breaks). **com.ibm.wmqfte.convertLineSeparators** For text transfers. Specifies whether the line separator sequences CRLF and LF are converted to the required line separator sequence for the destination. This conversion currently only takes effect for the following cases: • If the user-defined metadata key **com.ibm.wmqfte.newRecordOnLineSeparator** is set to `false` and the transfer is to a record-oriented file. • If the user-defined metadata key **com.ibm.wmqfte.com.ibm.wmqfte.insertRecordLineSeparator** is set to `false` and the transfer is from a record-oriented file. See also **fteCreateTransfer**: start a new file transfer.	false

Table 72. Advanced agent properties: General (continued)

Property name	Description	Default value
failTransferOnFirstFailure	The value of this property can be `true` or `false`. It allows an agent to be configured to fail a managed transfer as soon as a transfer item within that managed transfer fails. To enable this feature, APAR IT03450 must be applied for both the source agent and the destination agent, and the **`failTransferOnFirstFailure`** property must be set to `true` in the source agent's `agent.properties` file. Setting the property to `true` on the destination agent is optional. When the **`failTransferOnFirstFailure`** property is set to `true`, the agent starts processing managed transfer requests as normal. However, as soon as a transfer item fails, then the managed transfer is marked as failed and no further transfer items are processed. Transfer items that were successfully processed before the managed transfer failed are handled in the following way: • The source disposition for those transfer items is honored. For example, if the source disposition for the transfer item was set to `delete`, then the source file is deleted. • The destination files that were written remain on the destination file system and are not deleted. If the **`failTransferOnFirstFailure`** property is not set to `true` and a managed file transfer contains multiple files and one of these files fails to transfer, for example because the destination file already exists and the overwrite property is set to `error`, the source agent continues and attempts to transfer any remaining files in the request.	false
itemsPerProgressMessage	The number of files that are transferred before an agent publishes its next progress log message. This property controls the rate that progress log messages are published to the coordination queue manager during a transfer. The maximum value this property can be set to is 1000. **Note:** Progress messages include information about every file that is transferred since the last progress message was published. Increasing this value increases the size of the progress messages, which might affect performance.	50
maxInlineFileSize	For single file-to-file, or file-to-message transfers, the maximum file size (in bytes) that can be automatically included in the initial transfer request message. You can use this property to improve the speed of your transfers, but if you set the file size to too large a value, this might degrade performance. A suggested initial size for this property is 100 KB but you are recommended to thoroughly test different values until you find the best file size for your system. This feature is turned off by default, or by setting the **`maxInlineFileSize`** property to `0`.	0

Table 73. Advanced agent properties: High availability

Property name	Description	Default value
highlyAvailable	This property is read during agent startup and if set to the value `true`, the agent is started in high availability mode. If you do not specify the property, or set the value to false, the agent starts as a non-highly available agent.	false
standbyPollInterval	This property is used by the standby instance to attempt to open the shared queue at specified intervals. From IBM MQ 9.3.0, this property is also used by all instances to determine how long an instance waits between reconnection attempts if it becomes disconnected from its agent queue manager. The attempts are repeated until either an instance reconnects to its agent queue manager, opens the `SYSTEM.FTE.HA.<agent name>` queue (if it has already registered itself as a standby instance), or is stopped by the **`fteStopAgent`** command.	5 seconds
standbyStatusDiscardTime	This property sets the time duration for which the active instance waits for a status publication from a standby instance. If no publication is received from a standby instance, even after this ait time, the active instance removes the standby instance information from its list of standby instances. The default value is twice that of the **`standbyStatusPublishInterval`** property, so that the active instance waits longer before removing the standby instance from its list.	600 seconds

Table 73. Advanced agent properties: High availability (continued)		
Property name	**Description**	**Default value**
standbyStatusExpiry	This property sets the expiry time of the standby status message put to the command queue of an agent. The message expires if the active instance of an agent does not process this message.	30 seconds
standbyStatusPublishInterval	This property is used to set the frequency at which the standby instance publishes its state.	300 seconds

Table 74. Advanced agent properties: Input/output		
Property name	**Description**	**Default value**
doNotUseTempOutputFile	By default, the agent writes to a temporary file at the destination and renames this temporary file to the required file name after the file transfer is complete. Setting this value to true causes the agent to write directly to the final destination file. z/OS On z/OS systems, this behavior does not apply to sequential data sets, but does apply to PDS data set members. The value of this property for a transfer is defined by the destination agent.	false
enableMandatoryLocking	When accessing normal files, Managed File Transfer takes a shared lock for reading and an exclusive lock for writing. Windows On Windows file locking is advisory only. When this property is set to true, Managed File Transfer enforces file locking. On Windows this means that if another application has a file open, monitoring of that file does not trigger until the file is closed. Managed File Transfer transfers involving that file fail. UNIX On UNIX type platforms, file locking is fulfilled across processes. For UNIX type platforms, setting this property has no effect. This property applies to normal Managed File Transfer agents only. Managed File Transfer does not support the file locking mechanism on a file server. This property therefore does not work for a protocol bridge agent because protocol bridge agent does not lock a file on a file server when transferring a file. The value of this property can be `true` or `false`.	false
ioIdleThreadTimeout	Time in milliseconds for a file system input/output thread to remain idle before the thread shuts down. V 9.3.0 z/OS From IBM MQ 9.3.0, this property does not apply to agents running on IBM MQ for z/OS.	10000
ioQueueDepth	The maximum number of input/output requests to queue up.	10
ioThreadPoolSize	Maximum number of file system input/output threads available. Typically each transfer uses its own file system input/output thread, but if the number of concurrent transfers exceeds this limit, the file system input/output threads are shared between transfers. If you think you are likely to regularly have more concurrent transfers in progress than the ioThreadPoolSize value, you might see an improvement by increasing this value, so that each transfer has its own file system input/output thread.	10
textReplacementCharacterSequence	For text mode transfer, if any of the data bytes cannot be converted from the source code page to the destination code page, the default behavior is for the file transfer to fail. Set this property to allow the transfer to complete successfully by inserting the specified character value. This property value is a single character. Typically, a question mark (?) is used for any unmappable characters. For example, use this format textReplacementCharacterSequence=? where the question mark (?) is the replacement character. You cannot use a white space character as a replacement character.	None

V 9.3.0

Table 75. Advanced agent properties: Transfer log

Property name	Description	Default value
V 9.3.0 logTransfer See "Output produced by the LogTransfer function" on page 208 for examples of the logging information produced.	Turn on or turn off transfer logging. The possible values are: **info** Enables high level log information of a transfer. This is the default value. **moderate** Enables intermediate level log information of a transfer. **verbose** Enables detailed log information of a transfer. **off** Turns off transfer logging	info
V 9.3.0 logTransferFileSize	Defines the maximum size of a transfer log file in megabytes	20
V 9.3.0 logTransferFiles	Defines the maximum number of transfer files that are retained before the oldest file is discarded.	5

Table 76. Advanced agent properties: Multi-channel support

Property name	Description	Default value
agentMultipleChannelsEnabled	Setting this property to `true` enables a Managed File Transfer Agent to send transfer data messages across multiple IBM MQ channels. In some scenarios, setting this property might improve performance. However, only enable multi-channel support only if there is a demonstrable performance benefit. Only messages that are put to the SYSTEM.FTE.DATA.*destinationAgentName* queue are sent across multiple channels. The behavior for all other messages remains unchanged. When you set this property to `true`, you must also complete the IBM MQ configuration steps in one of the following topics to enable multi-channel support: • Configuring an MFT agent for multiple channels in a cluster • Configuring an MFT agent for multiple channels: non-clustered Additionally, you must also complete the standard IBM MQ configuration steps that are required for a Managed File Transfer agent, which are detailed in Configuring MFT for first use. The value of this property can be `true` or `false`.	false
agentMessageBatchSize	When configured with multiple channels, a source agent sends data messages for a transfer across each channel on a round-robin basis. This property controls the number of messages that are sent down each channel at a time.	5

Table 77. Advanced agent properties: Multi-instance queue manager

Property name	Description	Default value
agentQMgrStandby	The host name and the port number that are used for client connections, in IBM MQ CONNAME format, for the standby instance of a multi-instance agent queue manager that is defined by agentQMgr. For example, *host_name*(*port_number*) The agent attempts to connect to the standby queue manager when it detects a connection broken error, for example, MQRC 2009. Once the agent gets connected to the standby queue manager the agent remains connected until the standby queue manager becomes unavailable.	No default

Table 78. Advanced agent properties: Process controller

Property name	Description	Default value
agentQMgrRetryInterval	The interval, in seconds, between checks on the availability of the queue manager by the agent's process controller.	30
maxRestartCount	The maximum number of restarts that can happen within the time interval that is specified by the value of the maxRestartInterval property. When this value is exceeded the agent's process controller stops restarting the agent, and instead makes an action that is based on the value of the maxRestartDelay property.	4

Table 78. Advanced agent properties: Process controller (continued)

Property name	Description	Default value
maxRestartInterval	The interval, in seconds, that the agent's process controller measures agent restarts over. If the number of restarts in this interval exceeds the value of the maxRestartCount property, the agent's process controller stops restarting the agent. Instead the agent's process controller makes an action that is based on the value of the maxRestartDelay property.	120
maxRestartDelay	Determines the behavior of the agent's process controller when the rate of agent restarts exceeds the value of the maxRestartCount and maxRestartInterval properties. If you specify a value less than or equal to zero, the agent's process controller is stopped. If you specify a value greater than zero, it is the number of seconds to wait before the restart history information held by the agent's process controller is reset and the agent is restarted.	-1

Table 79. Advanced agent properties: Protocol bridge

Property name	Description	Default value
protocolBridgeCredentialConfiguration	The value of this property is passed in as a string to the initialize() method of the exit classes that are specified by protocolBridgeCredentialExitClasses.	null
protocolBridgeCredentialExitClasses	Specifies a comma-separated list of classes that implement a protocol bridge credential user exit routine. For more information, see Mapping credentials for a file server by using exit classes.	No default.
protocolBridgeDataTimeout	The timeout in milliseconds that the protocol bridge agent waits to either establish a data connection to an FTP server or to receive data from an FTP server over a connection that is already established. If you set this property to a value of 0, the protocol bridge agent waits indefinitely. If the timeout elapses, the protocol bridge agent closes any existing data connections to the FTP server and attempts to establish a new data connection before resuming the current transfer. If the attempt to establish the new data connection fails, the current transfer also fails.	0
protocolBridgeLogoutBeforeDisconnect	Specifies whether the protocol bridge agent logs the user out of the file server before closing the FTP session and disconnecting. If you set this property to `true`, the protocol bridge agent issues an FTP **`QUIT`** command to the file server.	false
protocolBridgePropertiesConfiguration	Passed as one of the bridge properties to the initialize() method of the exit classes that are specified by the protocolBridgeServerPropertiesExitClasses property.	No default
protocolBridgePropertiesExitClasses	Specifies a comma-separated list of classes that implement a protocol bridge server properties user exit routine. For more information, see ProtocolBridgePropertiesExit2: Looking up protocol file server properties.	No default

Table 80. Advanced agent properties: Protocol bridge agent logging

Property name	Description	Default value
agentLog	Key value pair component and operation to enable or disable logging of FTP commands and responses between the Protocol Bridge Agent and FTP/SFTP/FTPS file servers. For example: **`agentLog=on`** Turn on logging for all components **`agentLog=off`** Turn off logging for all components **`agentLog=ftp=on, sftp=on, ftps=off`** Turn on logging for FTP and SFTP, and turn off for FTPS	No default
agentLogFileSize	Defines the maximum size of a capture file in megabytes. Same as the default for regular trace default file size.	20

Table 80. Advanced agent properties: Protocol bridge agent logging (continued)		
Property name	**Description**	**Default value**
agentLogFiles	Defines the maximum number of capture files that are retained before the oldest file is discarded. V 9.3.0 • The default value of the **agentLogFiles** agent property has changed from 10 to 5. This means that from IBM MQ 9.3.0, if the default is set, there can be a maximum of five protocol bridge agent event log files, starting from `agentevent0.log` to `agentevent4.log`. However, you can change this value if required. • If the agent is migrated from a version prior to IBM MQ 9.3.0, you should manually delete the `agentevent5.log` to `agentevent9.log` files if any exist. • However, the size of each log file remains at 20 MB.	V 9.3.0 From IBM MQ 9.3.0, the default value is 5. Before IBM MQ 9.3.0, the default value is 10.

agentLogFilter	By default captures communication with all FTP servers the agent is connecting to.
	For example:
	• Filter on host/ip address
	`host=ftpprod.ibm.com, ftp2.ibm.com`
	`host=9.182.*`
	• Filter based on metadata
	`metadata="outbound files to xyz corp"`
		*

Table 81. Advanced agent properties: Queue		
Property name	**Description**	**Default value**
dynamicQueuePrefix	This property defines the prefix to use when creating a temporary dynamic queue.	WMQFTE.*
modelQueueName	This property defines the name of the module queue to use when creating a temporary dynamic queue.	SYSTEM.DEFAULT.MODEL.QUEUE
publicationMDUser	The MQMD user ID to associate with messages sent to be published by the coordination queue manager. If you do not set this property, the MQMD user ID is set based on the IBM MQ rules for setting MQMD user IDs.	No default

Table 82. Advanced agent properties: Resource monitoring		
Property name	**Description**	**Default value**
monitorFilepathPlatformSeparator	Specifies whether to use platform-specific path separators within the *$FILEPATH* variable. A value of `true` uses platform-specific path separators. A value of `false` uses a UNIX style forward slash (/) path separator on all platforms.	true
monitorMaxResourcesInPoll	Specifies the maximum number of monitored resources to be triggered in each poll interval. For example, if you specify a monitor pattern of `*.txt`, a poll interval of 10 seconds, and set the monitorMaxResourcesInPoll property to 10, the monitorMaxResourcesInPoll property limits the agent to trigger on a maximum of 10 matches for each poll interval. Matching resources beyond the limit of 10 are triggered in later poll intervals. In addition, you can use the monitorMaxResourcesInPoll property in combination with a matching **-bs** parameter on the **fteCreateMonitor** command, for example, to restrict each poll interval to triggering one transfer only. A value less than or equal to zero means that the number of monitor resources that are triggered in a polling interval is unlimited.	-1
monitorReportTriggerFail	Specifies whether failure conditions, in the environment and configuration, that are detected in the monitor are reported as a log message to the SYSTEM.FTE topic. A value of `true` logs messages. A value of `false` does not log messages.	true
monitorReportTriggerNotSatisfied	Specifies whether a non-satisfied trigger sends a log message to the SYSTEM.FTE topic that contains the details. A value of `true` logs messages. A value of `false` does not log messages.	false

Table 82. Advanced agent properties: Resource monitoring (continued)		
Property name	**Description**	**Default value**
monitorReportTriggerSatisfied	Specifies whether a satisfied trigger sends a log message to the SYSTEM.FTE topic that contains the details. A value of `true` logs messages. A value of `false` does not log messages.	false
monitorSilenceOnTriggerFailure	The number of consecutive failures of the resource monitor trigger before the failures are no longer reported.	5
monitorStopOnInternalFailure	The number of consecutive internal FFDC conditions of the resource monitor before the monitor changes its state to stop.	10

Table 83. Advanced agent properties: Root directory		
Property name	**Description**	**Default value**
commandPath	Specifies the set of paths that commands can be called by, using one of the following methods: • Agent Ant fte:call Ant task, fte:filecopy, or fte:filemove tasks • In an XML message passed to an agent, using one of the supported Managed File Transfer Agent command XML schemas (for example, managedCall or managedTransfer). For information about the valid syntax of the value of the commandPath property, see commandPath MFT property. **Important:** Take extreme care when you set this property because any command in one of the specified **commandPath**s can effectively be called from a remote client system that is able to send commands to the agent. For this reason, by default, when you specify a **commandPath**: • Any existing agent sandbox is configured by the agent when it starts up so that all **commandPath** directories are automatically added to the list of directories that have denied access for a transfer. • Any existing user sandboxes are updated when the agent starts up so that all the **commandPath** directories (and their sub-directories) are added as `<exclude>` elements to the `<read>` and `<write>` elements. • If the agent is not configured to use either an agent sandbox, or user sandboxes, then a new agent sandbox is created when the agent starts up that has the **commandPath** directories specified as denied directories. The value of this property can contain environment variables. See “The use of environment variables in MFT properties” on page 163 for more information. You can set the **addCommandPathToSandbox** property to false to override this default behavior. **Important:** Be aware that this override effectively enables a client to transfer any command to the agent system and call the command, and so should be used with extreme care.	None - no commands can be called
addCommandPathToSandbox	Specifies whether the directories specified by the **commandPath** property (and all of their sub directories) should be added to: • The denied directories for an existing agent sandbox. • The `<exclude>` elements for the `<read>` and `<write>` elements for any user sandboxes that have been defined. • A new agent sandbox, if an agent has not been configured with either an agent sandbox, or one or more user sandboxes. For information about the valid syntax of the value of the **commandPath** property, see commandPath MFT property.	True
additionalWildcardSandboxChecking	Specifies whether additional checks are to be made on wildcard transfers for an agent that has been configured with a user or agent sandbox in order to restrict the locations that the agent can transfer files to and from. When this property is set to true, the additional checking is enabled. If a transfer request attempts to read a location that is outside of the defined sandbox for file matching of the wildcard, the transfer fails. If there are multiple transfers within one transfer request, and one of these requests fails due to it attempting to read a location outside of the sandbox, the entire transfer fails. If checking fails, the reason for failure is given in an error messages (see Additional checks for wildcard transfers). If the property is omitted or set to false then no additional checks are made on wildcard transfers.	None

Table 83. Advanced agent properties: Root directory (continued)		
Property name	**Description**	**Default value**
sandboxRoot	Specifies the set of root paths to include and exclude when you use sandboxing. See Working with MFT agent sandboxes for information about this feature. Separate paths with a platform-specific path separator. Prefix paths with an exclamation point (!) character to denote paths as excluded from the sandbox. This feature is useful if you want to exclude a subdirectory under an included root path. The sandboxRoot property is not supported on protocol bridge agents. You cannot specify the sandboxRoot property and the userSandboxes property together. The value of this property can contain environment variables. See "The use of environment variables in MFT properties" on page 163 for more information.	None - no sandbox
transferRoot	Default root directory for relative paths that are specified to the agent. The value of this property can contain environment variables. See "The use of environment variables in MFT properties" on page 163 for more information.	The home directory for the user that started the agent process.
transferRootHLQ	Default HLQ (user ID) for non-fully qualified data sets specified to the agent	The user name of the user that started the agent process.
userSandboxes	Restrict the area of the file system that files can be transferred to and from based on the MQMD user name of the user that requests the transfer. For more information, see Working with MFT user sandboxes. The userSandboxes property is not supported on protocol bridge agents. You cannot specify the sandboxRoot property and the userSandboxes property together.	false

Table 84. Advanced agent properties: Scheduler property		
Property name	**Description**	**Default value**
maxSchedulerRunDelay	The maximum interval, in minutes, that the agent waits to check for scheduled transfers. Specify a positive integer to enable this property. For more information about why you might want to use this property, see What to do if your scheduled file transfer does not run or is delayed. Because the agent might be reading a command from its command queue at the time that scheduled transfers are due to run, there may be an additional delay before the scheduled transfers are started. In this case, the scheduler runs immediately after that command completes.	-1

Table 85. Advanced agent properties: Security		
Property name	**Description**	**Default value**
agentCredentialsKeyFile	Name of the file containing the credential key used while encrypting credentials.	A string property having no default value.
agentQMgrAuthenticationCredentialsFile	The path to the file that contains the credentials that should be used when connecting to the agent queue manager. The value of this property can contain environment variables. See "The use of environment variables in MFT properties" on page 163 for more information. ALW For details on creating the credentials files see MFT and IBM MQ connection authentication z/OS For details on creating the Authentication Credentials File see Configuring MQMFTCredentials.xml on z/OS.	The default value for this property is: z/OS Linux AIX `$HOME/MQMFTCredentials.xml` Windows `%USERPROFILE%/MQMFTCredentials.xml`
authorityChecking	Specifies whether the security features described in Restricting user authorities on MFT agent actions are enabled. The `inquire` permission is a required permission on all of the agent authority queues.	false
logAuthorityChecks	The level of authority check logging that is recorded in the agent event log in the `output0.log` file. The values that this property can have are `None` or `Failures` or `All`.	None

Table 85. Advanced agent properties: Security (continued)		
Property name	**Description**	**Default value**
userIdForClientConnect	The user ID that gets flowed through the client connections to IBM MQ. If *java* is specified, the user name reported by the JVM is flowed as part of the IBM MQ connection request. The values that this property can have are `None` or `java`.	None

Table 86. Advanced agent properties: SSL/TLS		
Property name	**Description**	**Default value**
agentSslCipherSpec	Specifies the protocol, hash algorithm, and encryption algorithm that is used, and how many bits are used in the encryption key, when data is exchanged between the agent and the agent queue manager. The value of agentSslCipherSpec is a CipherSpec name. This CipherSpec name is the same as the CipherSpec name used on the agent queue manager channel. A list of valid CipherSpec names is included in SSL/TLS CipherSpecs and CipherSuites in IBM MQ classes for Java and SSL/TLS CipherSpecs and CipherSuites in IBM MQ classes for JMS. agentSslCipherSpec is similar to agentSslCipherSuite. If both agentSslCipherSuite and agentSslCipherSpec are specified, the value of agentSslCipherSpec is used.	None
agentSslCipherSuite	Specifies SSL aspects of how the agent and the agent queue manager exchange data. The value of agentSslCipherSuite is a CipherSuite name. The CipherSuite name maps to the CipherSpec name used on the agent queue manager channel. For more information, see CipherSuite and CipherSpec name mappings. agentSslCipherSuite is similar to agentSslCipherSpec. If both agentSslCipherSuite and agentSslCipherSpec are specified, the value of agentSslCipherSpec is used.	None
agentSslPeerName	Specifies a distinguished name skeleton that must match the name that is provided by the agent queue manager. The distinguished name is used to check the identifying certificate that is presented by the queue manager on connection.	None
agentSslTrustStore	Specifies the location of the certificates that the agent trusts. The value of agentSslTrustStore is a file path. Parentheses, commas (,) and backslashes (\) are special characters in MFT commands, and must be escaped with a backslash (\) character. Windows File paths on Windows can be specified either using double backslashes (\\) as a separator, or using single forward slashes (/). The value of this property can contain environment variables.	None
agentSslKeyStore	Specifies the location of the private key of the agent. The value of agentSslKeyStore is a file path. Parentheses, commas (,) and backslashes (\) are special characters in MFT commands, and must be escaped with a backslash (\) character. Windows File paths on Windows can be specified either using double backslashes (\\) as a separator, or using single forward slashes (/). This property is only required if the agent queue manager requires client authentication. The value of this property can contain environment variables.	None
agentSslFipsRequired	Specifies that you want to enable FIPS support at the level of the agent. The value of this property can be `true` or `false`. For more information, see FIPS support in MFT.	false
agentSslKeyStoreType	The type of SSL keystore you want to use. JKS and PKCS#12 keystores are supported. The value of this property can be either `jks` or `pkcs12`.	jks
agentSslKeyStoreCredentialsFile	The path to the file that contains the credentials for accessing the key store of the agent. The value of this property can contain environment variables. See "The use of environment variables in MFT properties" on page 163 for more information. ALW For details on creating the credentials files see MFT and IBM MQ connection authentication z/OS For details on creating the Authentication Credentials File see Configuring MQMFTCredentials.xml on z/OS.	The default value for this property is: z/OS Linux AIX `$HOME/MQMFTCredentials.xml` Windows `%USERPROFILE%/MQMFTCredentials.xml`

Table 86. Advanced agent properties: SSL/TLS (continued)		
Property name	**Description**	**Default value**
agentSslTrustStoreType	The type of SSL keystore you want to use. JKS and PKCS#12 keystores are supported. The value of this property can be either `jks` or `pkcs12`.	jks
agentSslTrustStoreCredentialsFile	The path to the file that contains the credentials for accessing the trust store of the agent. The value of this property can contain environment variables. See "The use of environment variables in MFT properties" on page 163 for more information. ALW For details on creating the credentials files see MFT and IBM MQ connection authentication z/OS For details on creating the Authentication Credentials File see Configuring MQMFTCredentials.xml on z/OS.	The default value for this property is: z/OS Linux AIX `$HOME/MQMFTCredentials.xml` Windows `%USERPROFILE%/MQMFTCredentials.xml`

Table 87. Advanced agent properties: Timeout		
Property name	**Description**	**Default value**
maxTransferNegotiationTime	The maximum time in milliseconds that a transfer waits for a destination agent to complete negotiation. If negotiation does not complete in this time, the transfer is put into a resynchronization state and allows another transfer, when available, to run. In scenarios where the source or destination agent is under heavy load it is possible that the default value is too low for the agent to respond quickly enough to the negotiation request. This is most likely when a source agent has a large number of resource monitors defined or when its resource monitors are monitoring directories that contain large numbers of files. However, it can also occur when a large number of transfer requests is submitted to an agent. Increasing the value of this property to 200,000 or more may be necessary in such scenarios.	30 000
recoverableTransferRetryInterval	The time to wait in milliseconds between detecting a recoverable transfer error and attempting to resume the transfer.	60 000
senderTransferRetryInterval	The time in milliseconds to wait until a rejected transfer is retried because the destination is already running the maximum number of transfers. Minimum value is 1000.	30 000
transferAckTimeout	Timeout in milliseconds that a transfer waits for acknowledgment or data from the other end before a retry is issued. This is an advanced property and for most Managed File Transfer configurations it is not necessary to modify its value. Acknowledgments are sent from the receiving agent to the sending agent whenever a complete window of data is received. For bandwidth-constrained or unreliable networks and large agentWindowSize and agentChunkSize settings, it is possible that the default is not long enough. This can cause unnecessary retransfer of data between the agents. Therefore increasing this value might be beneficial and may reduce the likelihood of a transfer going into recovery mode because of a slow network.	60 000
transferAckTimeoutRetries	Maximum number of acknowledgment retries for a transfer without a response before the agent gives up and moves the transfer into a recovery state	5
xmlConfigReloadInterval	The interval in seconds between the agent reloading XML configuration files during runtime. To prevent the agent from reloading XML configuration files during runtime set this property to -1. The following XML configuration files are affected by this property: • ConnectDirectCredentials.xml • ConnectDirectNodeProperties.xml • ConnectDirectProcessDefinitions.xml • ProtocolBridgeCredentials.xml • ProtocolBridgeProperties.xml • UserSandboxes.xml	30

Table 88. Advanced agent properties: Tracing and logging

Property name	Description	Default value
javaCoreTriggerFile	The full path to a file location that the agent monitors. If the file exists at the specified location the agent startup will trigger a Javacore. After starting the agent, if you update a file at this location, the agent triggers a Javacore file again. A separate thread polls this file every 30 seconds to check whether the file has been created or updated. If the file has been created or updated since the last poll, the agent generates a Javacore file in the following directory: *MQ_DATA_PATH*/mqft/logs/*coordination_qmgr_name*/agents/*agent_name* When you specify this property, the agent outputs the following message at startup: `BFGAG0092I The <insert_0> file will be used to request JVM diagnostic information.` The value of this property can contain environment variables. See "The use of environment variables in MFT properties" on page 163 for more information.	None

trace	The trace specification to use when the agent is started. This is a comma-separated list of classes and/or packages, the equals character, and a trace level.
	For example, to trace the `com.ibm.wmqfte.agent.Agent` class and the classes in the `com.ibm.wmqfte.commandhandler` package from agent startup, add the following entry to the `agent.properties` file:
	`trace=com.ibm.wmqfte.agent.Agent,com.ibm.wmqfte.commandhandler=all`
	You can specify multiple trace specifications in a colon-separated list. For example,
	`trace=com.ibm.wmqfte.agent.Agent=all:com.ibm.wmqfte.commandhandler=moderate`
	The special trace specification `=all` is used to trace the agent and the Java Message Queuing Interface (JMQI) which handles all of the communication with the agent queue manager. To enable this, add the following entry to the `agent.properties` file:
	`trace==all`
	Unless otherwise specified by your IBM Support Representative, use the trace specification `com.ibm.wmqfte=all` like this:
	`trace=com.ibm.wmqfte=all`
		None

outputLogFiles	The total number of `output.log` files to keep. This value applies to an agent's process controller and the agent itself.	5
outputLogSize	The maximum size in MB of each `output.log` file before output wraps onto the next file. This value applies to an agent's process controller and the agent itself.	1
outputLogEncoding	The character encoding that the agent uses when it writes to the `output.log` file.	The default character encoding of the platform that the agent is running on.
traceFiles	The total number of trace files to keep. This value applies to an agent's process controller as well and the agent itself.	5
traceSize	The maximum size in MB of each trace file before trace wraps onto the next file. This value applies to an agent's process controller and the agent itself.	20
traceMaxBytes	The limit to the amount of message data that is output in the trace file.	4096 bytes
logTransferRecovery	When this property is set to a value of true, whenever a transfer enters recovery diagnostic events are reported to the agent's event log in the `output0.log` file.	For IBM MQ 9.0.0.0 and later, the default value is true.

Table 88. Advanced agent properties: Tracing and logging (continued)

Property name	Description	Default value
logCapture	Captures transfer request messages that are submitted to this agent and log messages that are published by the agent to the coordination queue manager. These captured messages can be helpful when debugging transfer problems. Captured messages are stored in files in the agent log directory called `capture?.log`. The ? is a numeric value. The file that contains the number `0` holds the newest captured messages.	false
logCaptureFileSize	Defines the maximum size of a capture file in megabytes.	10
logCaptureFiles	Defines the maximum number of capture files that are retained before the oldest file is discarded.	10
logCaptureFilter	A Java regular expression that the agent uses to match the topic name of the message. Only those messages that match the regular expression are captured.	.* (match all)
resourceMonitorLog	Key value pair of resource monitor and operation to turn on, or turn off, logging. The possible values are: • info • moderate • verbose • off For example: • `resourceMonitorLog=MON1,MON2=info:MON3=off` Turn on logging for `MON1` and `MON2`, and turn off logging for `MON3`. • `resourceMonitorLog=info` Turn on `info` level logging for all resource monitors. The resource monitor logs are written to a file named `resmonevent`*N*`.log`, where *N* stands for a number; for example, `resmonevent0.log`. **Attention:** All resource monitors of an agent write to the same log file. See Logging MFT resource monitors for more information.	info
resourceMonitorLogFileSize	Defines the maximum size of a capture file in megabytes.	20
resourceMonitorLogFiles	Defines the maximum number of capture files that are retained before the oldest file is discarded. V 9.3.0 • The default value of the **`resourceMonitorLogFiles`** agent property has changed from 10 to 5. This means that, from IBM MQ 9.3.0, if the default is set, there can be a maximum of five resource monitor event log files, starting from `resmonevent0.log` to `resmonevent4.log`. However, you can change this value if required. • If the agent is migrated from a version prior to IBM MQ 9.3.0, you should manually delete the `resmonevent5.log` to `resmonevent9.log` files if any exist. • However, the size of each log file remains at 20 MB.	V 9.3.0 From IBM MQ 9.3.0, the default value is 5. Before IBM MQ 9.3.0, the default value is 10.

Table 89. Advanced agent properties: Transfer limit

Property name	Description	Default value
maxDestinationTransfers	The maximum number of concurrent transfers that the destination agent processes at any point in time. Each transfer request that is submitted to an agent counts against this total regardless of the number of files that are transferred to satisfy the request. This means that a transfer request that transfers a single file counts in the same way as a transfer request that transfers 10 files. The agent queues transfers when the destination agent reaches the limit that is specified by the maxDestinationTransfers property. If the sum of the following agent property values: maxSourceTransfers + maxDestinationTransfers + maxQueuedTransfers exceeds the value of the MAXDEPTH setting of the state store queue (SYSTEM.FTE.STATE.*agent name*), the agent does not start.	25 (for all agents except Connect:Direct 5 (for Connect:Direct bridge agents)

Table 89. Advanced agent properties: Transfer limit (continued)

Property name	Description	Default value
maxFilesForTransfer	The maximum number of transfer items that are allowed for a single managed transfer. If a managed transfer contains more items than the value of maxFilesForTransfer, the managed transfer fails and no transfer items are processed. Setting this property prevents you from accidentally transferring too many files because of a bad transfer request, for example, if a user accidentally specifies the transfer of the root directory / on keyword conref="../common/mqent.dita#mqent/unixlinuxbis"/> systems.	5000
maxSourceTransfers	The maximum number of concurrent transfers that the source agent processes at any point in time. Each transfer request that is submitted to an agent counts against this total regardless of the number of files that are transferred to satisfy the request. This means that a transfer request that transfers a single file counts in the same way as a transfer request that transfers 10 files. The source agent queues transfers when the destination agent reaches the limit that is specified by the maxSourceTransfers property. If the sum of the following agent property values: maxSourceTransfers + maxDestinationTransfers + maxQueuedTransfers exceeds the value of the MAXDEPTH setting of the state store queue (SYSTEM.FTE.STATE.*agent name*), the agent does not start.	25 (for all agents except Connect:Direct bridge agents) 5 (for Connect:Direct bridge agents)
maxQueuedTransfers	The maximum number of pending transfers that can be queued by a source agent until the agent rejects a new transfer request. You can set this property so that despite of the limits of maxDestinationTransfers and maxSourceTransfers being met or exceeded, any new transfer requests that you make now are accepted, queued and then carried out later. The order that queued transfer requests are processed in is a factor of their priority and how long they have been queued. Old and high priority pending transfers are selected first. Transfers with a low priority that have been on the queue for a long time are selected in preference to newer, higher priority transfers. If the sum of the following agent property values: maxSourceTransfers + maxDestinationTransfers + maxQueuedTransfers exceeds the value of the MAXDEPTH setting of the state store queue (SYSTEM.FTE.STATE.*agent name*), the agent does not start.	1000

Table 90. Advanced agent properties: Transfer recovery timeout

Property name	Description	Default value
transferRecoveryTimeout	Set amount of time, in seconds, during which a source agent keeps trying to recover a stalled file transfer. When the property is not set, the default behavior of the agent is to keep retrying until it successfully recovers the transfer. You can set the following values for the transfer recovery timeout property: **-1** The agent continues to attempt to recover the stalled transfer until the transfer is complete. Using this option is the equivalent of the default behavior of the agent when the property is not set. **0** The agent stops the file transfer as soon as it enters recovery. **>0** The agent continues to attempt to recover the stalled transfer for the amount of time in seconds as set by the positive integer value specified. For example, `transferRecoveryTimeout=21600` indicates that the agent keeps trying to recover the transfer for 6 hours from when it enters recovery. The maximum value for this parameter is `999999999`.	-1

Table 91. Advanced agent properties: User exit routine

Property name	Description	Default value
agentForceConsistentPathDelimiters	Force the path delimiter in the source file and destination file information that is provided to the transfer exits to be the UNIX style: forward slash (/). Valid options are `true` and `false`.	false
destinationTransferEndExitClasses	Specifies a comma-separated list of classes that implement a destination transfer user exit routine.	No default
destinationTransferStartExitClasses	Specifies a comma-separated list of classes that implement a destination transfer start user exit routine.	No default

Table 91. Advanced agent properties: User exit routine (continued)		
Property name	**Description**	**Default value**
exitClassPath	Specifies a platform-specific, character-delimited list of directories that act as the class path for user exit routines. The agent's exits directory is searched before any entries in this class path.	Agent's `exits` directory
exitNativeLibraryPath	Specifies a platform-specific, character-delimited list of directories that act as the native library path for user exit routines.	Agent's `exits` directory
ioMaxRecordLength	The maximum record length, in bytes, that can be supported for a record-oriented file. Managed File Transfer can support writing to record-oriented files with any record length. However, large record lengths might cause out-of-memory errors, so to avoid these errors the maximum record length is restricted by default to 64 K. When reading from record-oriented files an entire record must fit into a single transfer chunk, therefore the record length is additionally limited by the transfer chunk size. This property is used only for I/O user exit record-oriented files.	64 KB
monitorExitClasses	Specifies a comma-separated list of classes that implement a monitor exit routine. For more information, see MFT resource monitor user exits.	No default
protocolBridgeCredentialExitClasses	Specifies a comma-separated list of classes that implement a protocol bridge credential user exit routine. For more information, see Mapping credentials for a file server by using exit classes.	No default.
sourceTransferEndExitClasses	Specifies a comma-separated list of classes that implement a source transfer end exit routine.	No default
sourceTransferStartExitClasses	Specifies a comma-separated list of classes that implement a source transfer start exit routine.	No default
IOExitClasses	Specifies a comma-separated list of classes that implement an I/O user exit routine. List only the classes that implement the IOExit interface, that is, do not list classes that implement the other I/O user exit interfaces, for example IOExitResourcePath and IOExitChannel. For more information, see Using MFT transfer I/O user exits.	No default.

Table 92. Advanced agent properties: IBM MQ client compression		
Property name	**Description**	**Default value**
agentDataCompression	This property is supported for client connections only. A comma-separated list of the compression types for the transfer of file data to negotiate with the remote IBM MQ server. You can find information about these compression types in the following topic: Message data compression list. The values are checked for validity and then passed through in order of appearance as properties to the agent client channel. The IBM MQ client then handles negotiation between this client channel and the remote server channel to find the matching lowest common denominator between the compression properties on the two channels. If no match is found, MQCOMPRESS_NONE is always selected.	MQCOMPRESS_NONE
agentHeaderCompression	This property is supported for client connections only. A comma-separated list of the compression types for the transfer of header data to negotiate with the remote IBM MQ server. Accepted values are MQCOMPRESS_NONE or MQCOMPRESS_SYSTEM. You can find information about these compression types in the following topic: HdrCompList [2] (MQLONG). The values are checked for validity and then passed through in order of appearance as properties to the agent client channel. The IBM MQ client then handles negotiation between this client channel and the remote server channel to find the matching lowest common denominator between the compression properties on the two channels. If no match is found, MQCOMPRESS_NONE is always selected.	MQCOMPRESS_NONE

z/OS

Table 93. Advanced agent properties: z/OS-specific

Property name	Description	Default value
adminGroup	A security manager group. Members of this group can: • Start the agent by using the **fteStartAgent** command. • Stop the agent by using the **fteStopAgent** command. • Turn on or turn off trace for the agent by using the **fteSetAgentTraceLevel** command. • Turn on or turn off logs for the agent by using the **fteSetAgentLogLevel** command • Display details of a local agent by running the **fteShowAgentDetails** command with the **-d** parameter specified. Define a security manager group, for example *MFTADMIN* and then add the started task userid and administrator TSO ids to this group. Edit the agent properties file and set the **adminGroup** property to be the name of this security manager group. `adminGroup=MFTADMIN`	None
bpxwdynAllocAdditionalOptions	Managed File Transfer uses the BPXWDYN text interface to create and open z/OS data sets. When BPXWDYN is used for data set allocation by default Managed File Transfer ensures, when possible, the data device is mounted (not required for disk-based data sets but is required for tape data sets). Because the options might not be supported for certain environments, use this property to change this behavior. Also when transferring to a data set it is also possible to specify options for BPXWDYN on the command line; these options are in addition to those options specified by this property. Some BPXWDYN options must not be specified when using the **bpxwdynAllocAdditionalOptions** property in the `agent.properties` file. For a list of these properties, see BPXWDYN properties you must not use with MFT.	Default is as follows: • MOUNT for z/OS V1R8 and later
armELEMTYPE	Optional property. If the agent is configured for restart by the Automatic Restart Manager (ARM), then set this property to the ARM ELEMTYPE parameter value specified in the associated ARM policy. For an agent, set ELEMTYPE to SYSBFGAG.	Not set
armELEMENT	Optional property. If the agent is configured for restart by the Automatic Restart Manager (ARM), then set this property to the ARM ELEMENT parameter value specified in the associated ARM policy. You can set the ELEMENT value to correspond to the agent name.	Not set

Table 94. Advanced agent properties: Other properties

Property name	Description	Default value
z/OS z/OS legacyXMLMessageMQMDFormat	Managed File Transfer XML messages that are generated by the agent (for example, log and transfer progress messages), are now sent to a queue with a blank MQMD format field. Previous versions of the product set the MQMD format field to MQSTR (a text message string). Setting this property to true enables the Managed File Transfer XML messages generated by the agent to be sent to a queue with MQMD format field of MQSTR. **Note:** Agent reply messages to commands will be sent with a message format matching the corresponding command request. If the MQMD format field is set to MQSTR, there is potential for Managed File Transfer command XML messages to be corrupted if there are channels in the MQ network with data conversion enabled.	false
adjustScheduleTimeForDaylightSaving	If your enterprise runs scheduled transfers every day, because the scheduled transfer was created with: • **-oi** parameter set to days, and • **-tb** parameter set to source on the fteCreateTransfer command for example, then setting this property to *true* will move the scheduled transfer time forward one hour when the clocks go forward by one hour and back by one hour when the clocks go back one hour. For example, if your scheduled transfer is due to run at 1:00 am, when the clocks go forward, the transfer will run at 2:00 am and when the clocks go back, the transfer reverts to 1:00 am.	true

Related concepts

MFT configuration options on Multiplatforms

Timeout option for file transfers in recovery

MFT sandboxes

Related tasks

Configuring an MFT agent for multiple channels in a cluster

Configuring an MFT agent for multiple channels: non-clustered

Related reference

"Java system properties for MFT" on page 211
A number of Managed File Transfer command and agent properties must be defined as Java system properties, because they define configuration for early function that is unable to use the command or agent properties mechanism.

SSL/TLS properties for MFT

"The MFT command.properties file" on page 195
The `command.properties` file specifies the command queue manager to connect to when you issue commands and the information that Managed File Transfer requires to contact that queue manager.

"The MFT coordination.properties file" on page 191
The `coordination.properties` file specifies the connection details to the coordination queue manager. Because several Managed File Transfer installations might share the same coordination queue manager, you can use a symbolic link to a common `coordination.properties` file on a shared drive.

"The MFT logger.properties file" on page 199
The Managed File Transfer logger has a set of configuration properties. Specify these properties in the `logger.properties` file, which is in the *MQ_DATA_PATH*`/mqft/config/`*coordination_qmgr_name*`/loggers/`*logger_name* directory.

fteCreateAgent

fteCreateBridgeAgent

fteCreateCDAgent

"The use of environment variables in MFT properties" on page 163
It is possible for environment variables to be used in Managed File Transfer properties that represent file or directory locations. This allows the locations of files or directories used when running parts of the product to vary depending on the current environment (such as the user running a command, for example).

The MFT coordination.properties file

The `coordination.properties` file specifies the connection details to the coordination queue manager. Because several Managed File Transfer installations might share the same coordination queue manager, you can use a symbolic link to a common `coordination.properties` file on a shared drive.

The `coordination.properties` file is created by the installer or by the **fteSetupCoordination** command. You can use the **fteSetupCoordination** command with the **-f** flag to change the basic coordination queue manager properties in this file. To change or add advanced coordination queue manager properties you must edit the file in a text editor.

The `coordination.properties` file is located in the *MQ_DATA_PATH*`/mqft/config/`*coordination_qmgr_name* directory.

The MFT `coordination.properties` file contains the following values:

Table 95. Basic coordination queue manager properties

Property name	Description	Default value
coordinationCredentialsKeyFile	Name of the file containing the credential key used while encrypting credentials.	A string property having no default value.
coordinationQMgr	The name of the coordination queue manager.	No default
coordinationQMgrHost	The host name or IP address of the coordination queue manager.	No default

Table 95. Basic coordination queue manager properties (continued)		
Property name	**Description**	**Default value**
coordinationQMgrPort	The port number used for client connections to the coordination queue manager.	1414
coordinationQMgrChannel	The SVRCONN channel name used to connect to the coordination queue manager.	SYSTEM.DEF.SVRCONN

If you do not specify a value for the coordinationQMgrHost property, bindings mode is used by default.

If you specify a value for the coordinationQMgrHost property but do not specify values for the coordinationQMgrPort and coordinationQMgrChannel properties, a port number of 1414 and a channel of SYSTEM.DEF.SVRCONN are used by default.

Here is an example of the contents of a `coordination.properties` file:

```
coordinationQMgr=ERIS
coordinationQMgrHost=kuiper.example.com
coordinationQMgrPort=2005
coordinationQMgrChannel=SYSTEM.DEF.SVRCONN
```

In this example, ERIS is the name of an IBM MQ queue manager that is located on the system `kuiper.example.com`. The queue manager ERIS is the queue manager that Managed File Transfer sends log information to.

Advanced coordination properties

Managed File Transfer also provides more advanced coordination properties. If you want to use any of the following properties, manually edit the `coordination.properties` file to add the required advanced properties. Parentheses, commas (,) and backslashes (\) are special characters in MFT commands, and must be escaped with a backslash (\) character. Windows File paths on Windows can be specified either using double backslashes (\\) as a separator, or using single forward slashes (/). For more information about character escaping in Java properties files, see the Oracle documentation Javadoc for the Properties class.

- Agent properties
- Code page properties
- Connection properties
- Multi-instance queue manager properties
- Queue properties
- Security properties
- SSL properties
- Subscription properties

Table 96. Advanced coordination properties: Agent		
Property name	**Description**	**Default value**
agentStatusJitterTolerance	The maximum amount of time an agent status message publication can be delayed by before the message is considered as overdue. This value is measured in milliseconds. The age of a status message is based on the time at which it was published at the coordination queue manager. However, the message is emitted by the agent some time before it is received at the coordination queue manager to allow for the time required to travel across the IBM MQ network. If this transit always takes the same amount of time, messages created 60 seconds apart are published 60 seconds apart, regardless of the actual time in transit. However, if the transit time varies between messages, they might be created at 60-second intervals but published at intervals of, for example, 61, 59, 58, and 62 seconds. The maximum deviation from 60, 2 seconds in this example, is the jitter. This property determines the maximum delay due to jitter before the message is treated as overdue.	3000

Table 97. Advanced coordination properties: Code page

Property name	Description	Default value
coordinationCcsid	The code page the commands connect to the coordination queue manager with. Also any publications to the coordination queue manager made by the agent are performed with this code page. If you specify a value for coordinationCcsid you must also specify a value for coordinationCcsidName.	1208
coordinationCcsidName	The Java representation of the coordinationCcsid. If you specify a value for coordinationCcsidName you must also specify a value for coordinationCcsid.	UTF8

Table 98. Advanced coordination properties: Connection

Property name	Description	Default value
javaLibraryPath	When connecting to a queue manager in bindings mode Managed File Transfer must have access to the IBM MQ Java bindings libraries. By default Managed File Transfer looks for the bindings libraries in the default location defined by IBM MQ. If the bindings libraries are in a different location use this property to specify the location of the bindings libraries.	*MQ_INSTALLATION_PATH*`/java/lib`

Table 99. Advanced coordination properties: Multi-instance queue manager

Property name	Description	Default value
coordinationQMgrStandby	The host name and the port number used for client connections, in IBM MQ CONNAME format, for the standby instance of a multi-instance coordination queue manager defined by the coordinationQMgr property. For example, *host_name*(*port_number*)	No default

Table 100. Advanced coordination properties: Queue

Property name	Description	Default value
dynamicQueuePrefix	This property defines the IBM MQ prefix to use for generating a temporary queue name. The format of the dynamicQueuePrefix property follows the format of the **DynamicQName** field of the IBM MQ MQOD structure. For more information, see Creating dynamic queues. You can also define this property in the `command.properties` file if you want to use a specific IBM MQ prefix for temporary reply queues that are generated by commands that require a response from the agent.	WMQFTE.*
modelQueueName	This property defines the IBM MQ model queue to use for generating a temporary queue. You can also define this property in the command.properties file if you want to use a specific IBM MQ model queue for temporary reply queues that are generated by commands that require a response from the agent. For more information, see “The MFT command.properties file” on page 195.	SYSTEM.DEFAULT.MODEL.QUEUE

Table 101. Advanced coordination properties: Security

Property name	Description	Default value
userIdForClientConnect	The user ID that gets flowed through the client connections to IBM MQ. If *java* is specified the user name reported by the JVM is flowed as part of the IBM MQ connection request. The value of this property can be `None` or `java`.	None
coordinationQMgrAuthenticationCredentialsFile	The path to the file that contains the MQ connection credentials for connection to the coordination queue manager.	z/OS For details on creating the Authentication Credentials File see Configuring MQMFTCredentials.xml on z/OS. ALW See Configuring MQMFTCredentials.xml for information on the location and permissions of this file. ALW Further details on creating the Authentication Credentials File are in MFT and IBM MQ connection authentication

Table 102. Advanced coordination properties: SSL/TLS

Property name	Description	Default value
coordinationSslCipherSpec	Specifies the protocol, hash algorithm, and encryption algorithm that is used, and how many bits are used in the encryption key, when data is exchanged between the commands and the coordination queue manager. The value of coordinationSslCipherSpec is a CipherSpec name. This CipherSpec name is the same as the CipherSpec name used on the coordination queue manager channel. A list of valid CipherSpec names is included in SSL/TLS CipherSpecs and CipherSuites in IBM MQ classes for Java and SSL/TLS CipherSpecs and CipherSuites in IBM MQ classes for JMS. coordinationSslCipherSpec is similar to coordinationSslCipherSuite. If both coordinationSslCipherSuite and coordinationSslCipherSpec are specified, the value of coordinationSslCipherSpec is used.	None
coordinationSslCipherSuite	Specifies SSL aspects of how the commands and the coordination queue manager exchange data. The value of coordinationSslCipherSuite is a CipherSuite name. The CipherSuite name maps to the CipherSpec name used on the agent queue manager channel. For more information, see CipherSuite and CipherSpec name mappings. coordinationSslCipherSuite is similar to coordinationSslCipherSpec. If both coordinationSslCipherSuite and coordinationSslCipherSpec are specified, the value of coordinationSslCipherSpec is used.	None
coordinationSslPeerName	Specifies a distinguished name skeleton that must match the name that is provided by the coordination queue manager. The distinguished name is used to check the identifying certificate that is presented by the coordination queue manager on connection.	None
coordinationSslTrustStore	Specifies the location of the certificates that the commands trust. The value of coordinationSslTrustStore is a file path. Parentheses, commas (,) and backslashes (\) are special characters in MFT commands, and must be escaped with a backslash (\) character. Windows File paths on Windows can be specified either using double backslashes (\\) as a separator, or using single forward slashes (/). From IBM WebSphere MQ 7.5 or later, the value of this property can contain environment variables.	None
coordinationSslTrustStoreType	The type of SSL keystore you want to use. JKS and PKCS#12 keystores are supported. The value of this property can be either `jks` or `pkcs12`.	jks
coordinationSslTrustStoreCredentialsFile	The path to the file that contains the coordinationSslTrustStore credentials. The value of this property can contain environment variables.	The default value for this property is `%USERPROFILE%/MQMFTCredentials.xml` on Windows and `$HOME/MQMFTCredentials.xml` on other platforms.
coordinationSslKeyStore	Specifies the location of the private key of the commands. The value of coordinationSslKeyStore is a file path. Parentheses, commas (,) and backslashes (\) are special characters in MFT commands, and must be escaped with a backslash (\) character. Windows File paths on Windows can be specified either using double backslashes (\\) as a separator, or using single forward slashes (/). This property is only required if the coordination queue manager requires client authentication. The value of this property can contain environment variables.	None
coordinationSslKeyStoreType	The type of SSL keystore you want to use. JKS and PKCS#12 keystores are supported. The value of this property can be either `jks` or `pkcs12`.	jks
coordinationSslKeyStoreCredentialsFile	The path to the file that contains the coordinationSslKeyStore credentials. The value of this property can contain environment variables.	The default value for this property is `%USERPROFILE%/MQMFTCredentials.xml` on Windows and `$HOME/MQMFTCredentials.xml` on other platforms.
coordinationSslFipsRequired	Specifies that you want to enable FIPS support at the level of the coordination queue manager. The value of this property can be `true` or `false`. For more information, see FIPS support in MFT.	false

Table 103. Advanced coordination properties: Subscription		
Property name	**Description**	**Default value**
coordinationSubscriptionTopic	Use this property to specify a topic other than SYSTEM.FTE to subscribe to in order to obtain publications about the status of the IBM MQ network. All tooling still publishes to the SYSTEM.FTE topic, but you can change your IBM MQ topology to distribute these publications to different topics based on their content. You can then use this function to force the tooling to subscribe to one of these other topics.	SYSTEM.FTE

Related concepts

MFT configuration options on Multiplatforms

Related reference

fteSetupCoordination

SSL/TLS properties for MFT

"The MFT agent.properties file" on page 169

Each Managed File Transfer Agent has its own properties file, `agent.properties`, that must contain the information that an agent uses to connect to its queue manager. The `agent.properties` file can also contain properties that alter the behavior of the agent.

"The MFT command.properties file" on page 195

The `command.properties` file specifies the command queue manager to connect to when you issue commands and the information that Managed File Transfer requires to contact that queue manager.

"The MFT logger.properties file" on page 199

The Managed File Transfer logger has a set of configuration properties. Specify these properties in the `logger.properties` file, which is in the *MQ_DATA_PATH*`/mqft/config/`*coordination_qmgr_name*`/loggers/`*logger_name* directory.

The MFT command.properties file

The `command.properties` file specifies the command queue manager to connect to when you issue commands and the information that Managed File Transfer requires to contact that queue manager.

The `command.properties` file is created by the installer or by the **fteSetupCommands** command. You can use the **fteSetupCommands** command with the **-f** flag to change the basic command queue manager properties in this file. To change or add advanced command queue manager properties you must edit the file in a text editor.

Some Managed File Transfer commands connect to the agent queue manager or coordination queue manager instead of the command queue manager. For information about which commands connect to which queue manager, see Which MFT command connects to which queue manager.

The `command.properties` file is located in the *MQ_DATA_PATH*`/mqft/config/`*coordination_qmgr_name* directory.

The MFT `command.properties` file contains the following values:

Table 104. Basic command queue manager properties		
Property name	**Description**	**Default value**
connectionCredentialsKeyFile	Name of the file containing the credential key used while encrypting credentials.	A string property having no default value.
connectionQMgr	The name of the queue manager used to connect to the IBM MQ network.	No default
connectionQMgrHost	The host name or IP address of the connection queue manager.	No default
connectionQMgrPort	The port number used to connect to the connection queue manager in client mode.	1414
connectionQMgrChannel	The SVRCONN channel name used to connect to the connection queue manager.	SYSTEM.DEF.SVRCONN

If you do not specify a value for the connectionQMgrHost property, bindings mode is used by default.

If you specify a value for the connectionQMgrHost property but do not specify values for the connectionQMgrPort and connectionQMgrChannel properties, a port number of 1414 and a channel of SYSTEM.DEF.SVRCONN are used by default.

Here is an example of the contents of a `command.properties` file:

```
connectionQMgr=PLUTO
connectionQMgrHost=kuiper.example.com
connectionQMgrPort=1930
connectionQMgrChannel=SYSTEM.DEF.SVRCONN
```

In this example, PLUTO is the name of an IBM MQ queue manager that is located on the system `kuiper.example.com`. The queue manager PLUTO is the queue manager that the Managed File Transfer commands connect to.

Advanced command properties

Managed File Transfer also provides more advanced command properties. If you want to use any of the following properties, manually edit the `command.properties` file to add the required advanced properties. Parentheses, commas (,) and backslashes (\) are special characters in MFT commands, and must be escaped with a backslash (\) character. Windows File paths on Windows can be specified either using double backslashes (\\) as a separator, or using single forward slashes (/). For more information about character escaping in Java properties files, see the Oracle documentation Javadoc for the Properties class.

- Agent properties
- Code page properties
- Multi-instance queue manager properties
- Queue properties
- Security properties
- SSL properties

Table 105. Advanced command properties: Agent

Property name	Description	Default value
failCleanAgentWithNoArguments	By default, the value of this property is true, which means that the **fteCleanAgent** command fails to run if only the agent name parameter is specified. Setting the property to false means that, if only the agent name parameter is set, the behavior of the **fteCleanAgent** command is equivalent to specifying the **-all** parameter.	true

Table 106. Advanced command properties: Code page

Property name	Description	Default value
connectionCcsid	The code page the commands connect to the command queue manager with. If you specify a value for connectionCcsid you must also specify a value for connectionCcsidName.	1208
connectionCcsidName	The Java representation of the connectionCcsid. If you specify a value for connectionCcsidName you must also specify a value for connectionCcsid.	UTF8

Table 107. Advanced connection properties: Multi-instance queue manager

Property name	Description	Default value
connectionQMgrStandby	The host name and the port number used for client connections, in IBM MQ CONNAME format, for the standby instance of a multi-instance command queue manager defined by the connectionQMgr property. For example, *host_name*(*port_number*)	No default

Table 108. Advanced command properties: Queue		
Property name	**Description**	**Default value**
dynamicQueuePrefix	For commands that require a response from the agent, this property defines the IBM MQ prefix to use for generating the temporary reply queue name. The format of the dynamicQueuePrefix property follows the format of the **DynamicQName** field of the IBM MQ MQOD structure. For more information, see Creating dynamic queues. You can also define this property in the `coordination.properties` file if you want to use a specific IBM MQ prefix for temporary queues that are generated by WMQFTE.	WMQFTE.*
modelQueueName	For commands that require a response from the agent, this property defines the IBM MQ model queue to use for generating the temporary reply queue. You can also define this property in the coordination.properties file if you want to use a specific IBM MQ model queue for temporary queues that are generated by WMQFTE. For more information, see "The MFT coordination.properties file" on page 191.	SYSTEM.DEFAULT.MODEL.QUEUE
Connection properties:		
javaLibraryPath	When connecting to a queue manager in bindings mode Managed File Transfer must have access to the IBM MQ Java bindings libraries. By default Managed File Transfer looks for the bindings libraries in the default location defined by IBM MQ. If the bindings libraries are in a different location use this property to specify the location of the bindings libraries.	`/opt/mqm/java/lib`
z/OS z/OS legacyXMLMessageMQMDFormat	Managed File Transfer command XML messages are now sent to a queue with a blank MQMD format field. Previous versions of the product set the MQMD format field to MQSTR (a text message string). Setting this property to true enables the Managed File Transfer command XML messages to be sent to a queue with MQMD format field of MQSTR. If the MQMD format field is set to MQSTR, there is potential for Managed File Transfer command XML messages to be corrupted if there are channels in the MQ network with data conversion enabled.	false

Table 109. Advanced command properties: Security		
Property name	**Description**	**Default value**
userIdForClientConnect	The user ID that gets flowed through the client connections to IBM MQ. If *java* is specified the user name reported by the JVM is flowed as part of the IBM MQ connection request. The value of this property can be None or `java`.	None
connectionQMgrAuthenticationCredentialsFile	The path to the file that contains the MQ connection credentials for connection to the command queue manager.	See MFT and IBM MQ connection authentication and its child topics.

Table 110. Advanced command properties: SSL/TLS		
Property name	**Description**	**Default value**
connectionSslCipherSpec	Specifies the protocol, hash algorithm, and encryption algorithm that is used, and how many bits are used in the encryption key, when data is exchanged between the commands and the command queue manager. The value of connectionSslCipherSpec is a CipherSpec name. This CipherSpec name is the same as the CipherSpec name used on the command queue manager channel. A list of valid CipherSpec names is included in SSL/TLS CipherSpecs and CipherSuites in IBM MQ classes for Java and SSL/TLS CipherSpecs and CipherSuites in IBM MQ classes for JMS. connectionSslCipherSpec is similar to connectionSslCipherSuite. If both connectionSslCipherSuite and connectionSslCipherSpec are specified, the value of connectionSslCipherSpec is used.	None
connectionSslCipherSuite	Specifies SSL aspects of how the commands and the command queue manager exchange data. The value of connectionSslCipherSuite is a CipherSuite name. The CipherSuite name maps to the CipherSpec name used on the agent queue manager channel. For more information, see CipherSuite and CipherSpec name mappings. connectionSslCipherSuite is similar to connectionSslCipherSpec. If both connectionSslCipherSuite and connectionSslCipherSpec are specified, the value of connectionSslCipherSpec is used.	None

Table 110. Advanced command properties: SSL/TLS (continued)		
Property name	**Description**	**Default value**
connectionSslPeerName	Specifies a distinguished name skeleton that must match the name that is provided by the command queue manager. The distinguished name is used to check the identifying certificate that is presented by the command queue manager on connection.	None
connectionSslTrustStore	Specifies the location of the certificates that the commands trust. The value of connectionSslTrustStore is a file path. Parentheses, commas (,) and backslashes (\) are special characters in MFT commands, and must be escaped with a backslash (\) character. Windows File paths on Windows can be specified either using double backslashes (\\) as a separator, or using single forward slashes (/). The value of this property can contain environment variables.	None
connectionSslTrustStoreType	The type of SSL truststore you want to use. JKS and PKCS#12 keystores are supported. The value of this property can be either `jks` or `pkcs12`.	jks
connectionSslTrustStoreCredentialsFile	The path to the file that contains the connectionSslTrustStore credentials. The value of this property can contain environment variables.	The default value for this property is `%USERPROFILE%/MQMFTCredentials.xml` on Windows and `$HOME/MQMFTCredentials.xml` on other platforms.
connectionSslKeyStore	Specifies the location of the private key of the commands. The value of connectionSslKeyStore is a file path. Parentheses, commas (,) and backslashes (\) are special characters in MFT commands, and must be escaped with a backslash (\) character. Windows File paths on Windows can be specified either using double backslashes (\\) as a separator, or using single forward slashes (/). This property is only required if the command queue manager requires client authentication. The value of this property can contain environment variables.	None
connectionSslKeyStoreType	The type of SSL keystore you want to use. JKS and PKCS#12 keystores are supported. The value of this property can be either `jks` or `pkcs12`. The value of this property can contain environment variables.	jks
connectionSslKeyStoreCredentialsFile	The path to the file that contains the connectionSslKeyStore credentials. The value of this property can contain environment variables.	The default value for this property is `%USERPROFILE%/MQMFTCredentials.xml` on Windows and `$HOME/MQMFTCredentials.xml` on other platforms.
connectionSslFipsRequired	Specifies that you want to enable FIPS support at the level of the command queue manager. The value of this property can be `true` or `false`. For more information, see FIPS support in MFT.	false

Related concepts

MFT configuration options on Multiplatforms

Related reference

“Java system properties for MFT” on page 211

A number of Managed File Transfer command and agent properties must be defined as Java system properties, because they define configuration for early function that is unable to use the command or agent properties mechanism.

SSL/TLS properties for MFT

“The MFT agent.properties file” on page 169

Each Managed File Transfer Agent has its own properties file, `agent.properties`, that must contain the information that an agent uses to connect to its queue manager. The `agent.properties` file can also contain properties that alter the behavior of the agent.

“The MFT coordination.properties file” on page 191

The `coordination.properties` file specifies the connection details to the coordination queue manager. Because several Managed File Transfer installations might share the same coordination queue manager, you can use a symbolic link to a common `coordination.properties` file on a shared drive.

“The MFT logger.properties file” on page 199

The Managed File Transfer logger has a set of configuration properties. Specify these properties in the `logger.properties` file, which is in the *MQ_DATA_PATH*`/mqft/config/`*coordination_qmgr_name*`/loggers/`*logger_name* directory.

fteSetupCommands: create the MFT command.properties file

fteCleanAgent: clean up an MFT Agent

The MFT logger.properties file

The Managed File Transfer logger has a set of configuration properties. Specify these properties in the `logger.properties` file, which is in the *MQ_DATA_PATH*`/mqft/config/`*coordination_qmgr_name*`/loggers/`*logger_name* directory.

You can use environment variables in some Managed File Transfer properties that represent file or directory locations. This allows the locations of files or directories that are used when running parts of the product, to vary depending on environment changes, such as which user is running the process. For more information, see "The use of environment variables in MFT properties" on page 163.

Note: Parentheses, commas (,) and backslashes (\) are special characters in MFT commands, and must be escaped with a backslash (\) character. Windows File paths on Windows can be specified either using double backslashes (\\) as a separator, or using single forward slashes (/). For more information about character escaping in Java properties files in Oracle, see Javadoc for the Properties class.

The MFT `logger.properties` file contains the following values:

- "Bindings mode connection properties" on page 199
- "Client mode SSL/TLS connection properties" on page 206

Bindings mode connection properties

Table 111. Bindings mode connection properties for the logger.properties file

Property name	Description	Default value
wmqfte.logger.type	The logger type in use: file, or database. Set this value to `FILE`, or `DATABASE`.	No default value
wmqfte.max.transaction.messages	The maximum number of messages that is processed in a transaction before the transaction is committed. In circular logging mode, a queue manager has a fixed amount of space available for inflight data. Ensure that you set this property with a sufficiently low value so that the available space does not run out.	50
wmqfte.max.transaction.time	The maximum length of time in milliseconds that passes between transaction commits.	5000
wmqfte.max.consecutive.reject	The maximum number of messages that can be rejected consecutively (that is, without encountering a valid message). If this number is exceeded the logger concludes that the problem is not with the messages themselves but with the configuration. For example, if you make an agent-name column in the database narrower than all of your agent names, all messages referring to agents are rejected.	50
wmqfte.reject.queue.name	The name of a queue to which the logger puts messages that the logger cannot handle. If you have a database logger see MFT logger error handling and rejection for details of which messages might be put onto this queue.	SYSTEM.FTE.LOG.RJCT.*logger_name*
wmqfte.command.queue.name	The name of a queue that the logger reads command messages controlling its behavior from.	SYSTEM.FTE.LOG.CMD.*logger_name*

Table 111. Bindings mode connection properties for the logger.properties file (continued)

Property name	Description	Default value
wmqfte.queue.manager	The queue manager that the logger connects to. This parameter is required, and is all that is needed for bindings mode connections to the queue manager. (For the properties for connecting to a remote queue manager, see Table 112 on page 206.)	No default value
wmqfte.message.source.type	One of the following values: **`automatic subscription`** The default value. The logger creates and uses its own durable, managed subscription on the queue manager that is defined in SYSTEM.FTE/Log/#. This is an appropriate value for most scenarios. **`administrative subscription`** If the automatic subscription is not appropriate, you can define a different subscription (for example, by using the IBM MQ Explorer, MQSC, or PCF) and instruct the logger to use that subscription. For example, use this value to partition the log space so that one logger handles agents from A-H, another logger handles I-P, and a third logger from Q-Z. **queue** If the IBM MQ topology means that creating a subscription for the logger is not convenient, you can use a queue instead. Configure IBM MQ so that the queue receives the messages that are typically received by a subscription to SYSTEM.FTE/Log/# on the coordination queue manager.	automatic subscription
wmqfte.message.source.name	If the message source type is `administrative subscription` or queue, the name of the subscription or queue to use. This property is ignored if the source type is `automatic subscription`.	No default value
wmqfte.database.credentials.file	The file that contains the user name and password for connecting to the database. The value of this property can contain environment variables. For more information, see MFT credentials file format.	z/OS For information about creating the authentication credentials file, see Configuring MQMFTCredentials.xml on z/OS. ALW For information on the location and permissions of this file, see Configuring MQMFTCredentials.xml. ALW See also MFT and IBM MQ connection authentication.

Table 111. Bindings mode connection properties for the logger.properties file (continued)

Property name	Description	Default value
wmqfte.database.driver	The location of the JDBC driver classes for the database. This is typically the path and file name of a JAR file. AIX For example, the Type 2 driver for Db2 on AIX systems requires the file `/opt/IBM/db2/V9.5/java/db2jcc.jar`. Windows On Windows systems, specify the path separator as a forward slash character (/) for example, `C:/Program Files/IBM/SQLLIB/java/db2jcc.jar`. z/OS On z/OS, specify the full path of the `db2jcc.jar` file. For example, `wmqfte.database.driver=/db2/db2v10/jdbc/classes/db2jcc.jar`. z/OS On z/OS systems, you must reference all of the following JAR files: • `db2jcc.jar` • `db2jcc_license_cisuz.jar` • `db2jcc_javax.jar` If your database driver consists of multiple JAR files (for example, Db2 V9.1 requires a driver JAR file and a license JAR file), include all of these JAR files in this property. Separate multiple file names by using the classpath separator for your platform, that is, the semicolon character (;) on Windows systems and the colon character (:) on other platforms.	No default value
wmqfte.database.exclude .duplicate. metadata	Controls whether entries are stored in the metadata table that contains information that can be found in other tables within the database logger schema. Set this value to `true`, or `false`. These metadata entries are no longer stored by default as it is duplication of existing data and a waste of database storage capacity. The property entries and the tables, where the same data appears, are as follows: • com.ibm.wmqfte.SourceAgent TRANSFER_EVENT or CALL_REQUEST • com.ibm.wmqfte.DestinationAgent TRANSFER_EVENT • com.ibm.wmqfte.MqmdUser TRANSFER_EVENT or CALL_REQUEST • com.ibm.wmqfte.OriginatingUser TRANSFER_EVENT or CALL_REQUEST • com.ibm.wmqfte.OriginatingHost TRANSFER_EVENT or CALL_REQUEST • com.ibm.wmqfte.TransferId TRANSFER or CALL_REQUEST • com.ibm.wmqfte.JobName TRANSFER or CALL_REQUEST Setting the value of this property to `false` causes these metadata entries to be stored in the metadata table.	true

Table 111. Bindings mode connection properties for the logger.properties file (continued)

Property name	Description	Default value
wmqfte.database.host	Db2 only: The host name of the database server to connect to using a Type 4 JDBC driver. If a value for this property is specified, then a value for `wmqfte.database.port` must also be specified. If both properties are not defined, the database logger connects by using the default Type 2 JDBC driver. If a value for this property is specified, then a credentials file for this logger (file path defined by the wmqfte.database.credentials.file property) must exist, and be accessible to define the user name and password for connecting to the database, even if the database is on the local system.	No default value
wmqfte.database.name	The name of the database instance (or subsystem when using Db2 for z/OS) that contains the Managed File Transfer log tables.	No default value
wmqfte.database.type	The database management system in use: Db2 or Oracle. Set this value to db2 or `oracle`.	db2
wmqfte.database.port	Db2 only: The port number of the database server to connect to using a Type 4 JDBC driver. If a value for this property is specified, then a value for `wmqfte.database.host` must also be specified. If both properties are not defined, the database logger connects by using the default Type 2 JDBC driver. If a value for this property is specified, then a credentials file for this logger (file path defined by the wmqfte.database.credentials.file property) must exist, and be accessible to define the user name and password for connecting to the database, even if the database is on the local system.	No default value
wmqfte.database.schema	Db2 only: The database schema that contains the Managed File Transfer logging tables. In most cases the default value is appropriate, but you might need to specify an alternative value depending on your own site-specific database considerations.	FTELOG
wmqfte.database.native.library.path	The path that contains the native libraries that are needed by your chosen database driver (if any).  For example, the Type 2 driver for Db2 on AIX systems requires libraries from `/opt/IBM/db2/V9.5/lib32/`. As an alternative to this property, you can set the java.library.path system property by using other methods.	No default value
wmqfte.file.logger.fileDirectory	The directory where the file logger log files are located.	`mqft/logs/`*`coordination_dir`*`/loggers/`*`logger_name`*`/logs`
wmqfte.file.logger.fileSize	The maximum size that a log file is allowed to grow to. The size value is a positive integer, greater than zero, followed by one of the following units: KB, MB, GB, m (minutes), h (hours), d (days), w (weeks). For example, `wmqfte.file.logger.fileSize=5MB` specifies a maximum file size of 5MB, and `wmqfte.file.logger.fileSize=2d` specifies a maximum file size of 2 days of data.	10MB

Table 111. Bindings mode connection properties for the logger.properties file (continued)

Property name	Description	Default value
wmqfte.file.logger.fileCount	The maximum number of log files to create. When the amount of data exceeds the maximum amount that can be stored in this number of files, the oldest file is deleted so that the number of files never exceeds the value that is specified.	3
wmqfte.file.logger.mode	The logger mode in use: circular, or linear. Set this value to `CIRCULAR`, or `LINEAR`. CIRCULAR - The file logger writes information to a file until that file reaches its maximum size as defined by using the **wmqfte.file.logger.fileSize** property. When the maximum size is reached, the file logger starts a new file. The maximum number of files that are written in this mode is controlled by the value that is defined by using the **wmqfte.file.logger.fileCount** property. When this maximum number of files is reached, the file logger deletes the first file and re-creates it for use as the currently active file. If the value defined in the **wmqfte.file.logger.fileSize** property is a fixed size byte unit (for example, KB, MB, or GB) then the upper limit on disk space that is used in this mode equals fileSize multiplied by fileCount. If the value defined in the **wmqfte.file.logger.fileSize** property is a time unit (for example, `m`, h, d, or `w`) then the maximum size depends on the throughput of log messages in your system over these time periods. The log file naming convention that is used when running in this mode is:*`logger_name`*`number-`*`timestamp`*`.log` where: • *logger_name* is the name that is given to the logger in the **fteCreateLogger** command. • *number* is the number of the file within the set. • *timestamp* is the timestamp of when the file was created. For example, `LOGGER1-20111216123430147.log` LINEAR - The file logger writes information to a file until that file reaches its maximum size as defined by using the **wmqfte.file.logger.fileSize** property. When the maximum size is reached the file logger starts a new file. Previously written files are not deleted, which allows them to be kept as a historical record of log messages. Files are not deleted when running in `linear` mode, so the **wmqfte.file.logger.fileCount** property is ignored because there is no upper limit to the number of files that can be created. Because there is no upper limit when running in this mode, it is necessary to track the amount of disk space that is used by the log files to avoid running low on disk space. The log file naming convention that is used when running in this mode is: *`logger_name`*`-`*`timestamp`*`.log` where: • *logger_name* is the name that is given to the logger in the **fteCreateLogger** command. • *timestamp* is the timestamp of when the file was created. For example, `LOGGER-20111216123430147.log`	No default value

Table 111. Bindings mode connection properties for the logger.properties file (continued)

Property name	Description	Default value
wmqfte.max.retry.interval	The maximum time, in seconds, between retries when the logger encounters a persistent error. Some error conditions (for example, loss of database connection) prevent the logger from continuing. When this type of condition occurs, the logger rolls back the current transaction, waits for a period, and then retries. The time that the logger waits is initially very short, so that transitory errors can be overcome quickly. However, each time the logger retries, the time that it waits is increased. This prevents too much unnecessary work from taking place when the error condition is longer-lasting, for example when a database is taken down for maintenance. Use this property to set a limit to the length of the wait so that a retry occurs in a reasonable time of the error condition being resolved.	600
immediateShutdownTimeout	The time, in seconds, that the logger waits for any outstanding operations to complete and shut down gracefully. By default, the logger waits for 10 seconds for operations to complete. If operations are not completed before the timeout, the logger writes the following event message to `output0.log`, and ends. `BFGDB0082I: The logger is ending immediately.` If you specify the value of zero, the logger waits for ever to complete current operations. The default value is used if the value of **`immediateShutdownTimeout`** is set to less than zero. The property applies to both the stand-alone database logger as well as the file type logger.	10
loggerCredentialsKeyFile	Name of the file containing the credential key used while encrypting credentials.	A string property having no default value.
loggerQMgrRetryInterval	The interval, in seconds, between checks on the availability of the queue manager by the logger's process controller.	30
maxRestartCount	The maximum number of restarts that can happen within the time interval specified by the value of the maxRestartInterval property. When this value is exceeded the logger's process controller stops restarting the logger, and instead performs an action that is based on the value of the maxRestartDelay property.	4
maxRestartInterval	The interval, in seconds, that the logger's process controller measures logger restarts over. If the number of restarts in this interval exceeds the value of the maxRestartCount property, the logger's process controller stops restarting the logger. Instead the logger's process controller performs an action that is based on the value of the maxRestartDelay property.	120

Table 111. Bindings mode connection properties for the logger.properties file (continued)

Property name	Description	Default value
maxRestartDelay	Determines the behavior of the logger's process controller when the rate of logger restarts exceeds the value of the maxRestartCount and maxRestartInterval properties. If you specify a value less than or equal to zero, the logger's process controller is stopped. If you specify a value greater than zero, this is the number of seconds to wait before the restart history information held by the logger's process controller is reset and the logger is restarted.	-1
wmqfte.oracle.port	The port that the logger uses to connect to the Oracle instance. This port is also known as a TNS listener.	1521
wmqfte.oracle.host	The host that the logger uses to connect to the Oracle instance.	localhost
armELEMTYPE	Optional property. If the logger is configured for restart by the Automatic Restart Manager (ARM), then set this property to the ARM ELEMTYPE parameter value specified in the associated ARM policy. For a logger, set ELEMTYPE to SYSBFGLG.	Not set
armELEMENT	Optional property. If the logger is configured for restart by the Automatic Restart Manager (ARM), then set this property to the ARM ELEMENT parameter value specified in the associated ARM policy. You can set the ELEMENT value to correspond to the logger name.	Not set
loggerQMgrAuthenticationCredentialsFile	The path to the file that contains the MQ connection credentials for connection to the logger's coordination queue manager.	z/OS For information about creating the authentication credentials file, see Configuring MQMFTCredentials.xml on z/OS. ALW For information about the location and permissions for this file, see Configuring MQMFTCredentials.xml. ALW See also MFT and IBM MQ connection authentication.
trace	Optional property. Trace specification when the logger is to be run with trace enabled at logger start. The trace specification is a comma-separated list of classes, the equals character, and a trace level. For example, `com.ibm.wmqfte.databaselogger`, and `com.ibm.wmqfte.databaselogger.operation=all` You can specify multiple trace specifications in a colon-separated list. For example, `com.ibm.wmqfte.databaselogger=moderate:com.ibm.wmqfte.databaselogger.operation=all`	None
traceFiles	Optional property. The total number of trace files to keep. This value applies to the process controller of a logger, as well as the logger itself.	5
traceSize	Optional property. The maximum size in MB of each trace file, before trace wraps onto the next file. This value applies to the process controller of the logger, and the logger itself.	20

Table 111. Bindings mode connection properties for the logger.properties file (continued)		
Property name	**Description**	**Default value**
wmqfte.file.logger.filePermissions	Optional property. Use to specify what sort of permission is required for the log file of the logger. The property applies to both linear and circular logs, and can take the values *UserReadWriteOnly* or *UserReadWriteAllRead*. The *UserReadWriteOnly* value has the existing equivalent authority of 600 and the *UserReadWriteAllRead* value has the equivalent authority of 644. Any change in permission is applicable to newly created logger files. If you enter a value for the property that is not valid, the logger takes the default value and issues message BFGDB0083W to the output log.	UserReadWriteOnly

Client mode SSL/TLS connection properties

The properties required to support client mode connection to a logger queue manager using SSL/TLS.

Table 112. Client mode SSL/TLS connection properties for the logger.properties file		
Property name	**Description**	**Default value**
wmqfte.queue.manager.host	Host name, or IP address, of logger queue manager.	No default value
wmqfte.queue.manager.port	Port on which the logger queue manager is listening.	1414
wmqfte.queue.manager.channel	Name of the server connection channel on the logger queue manager.	SYSTEM.DEF.SVRCONN
wmqfte.Ssl.CipherSuite	Specifies TLS aspects of how the logger and the logger queue manager exchange data. The value of **wmqfte.Ssl.CipherSuite** is a CipherSuite name. The CipherSuite name maps to the CipherSpec name used on the logger queue manager channel. For more information, see CipherSuite and CipherSpec name mappings.	No default value
wmqfte.Ssl.PeerName	Specifies a distinguished name skeleton that must match the name that is provided by the logger queue manager. The distinguished name is used to check the identifying certificate that is presented by the queue manager on connection.	No default value
wmqfte.Ssl.TrustStore	Specifies the location of the certificates that the logger trusts. The value of **wmqfte.Ssl.TrustStore** is a file path. Parentheses, commas (,) and backslashes (\) are special characters in MFT commands, and must be escaped with a backslash (\) character. Windows File paths on Windows can be specified either using double backslashes (\\) as a separator, or using single forward slashes (/). Note that the value of this property can contain environment variables.	No default value

Table 112. Client mode SSL/TLS connection properties for the logger.properties file (continued)		
Property name	**Description**	**Default value**
wmqfte.Ssl.TrustStoreCredentialsFile	The path to the file that contains the **wmqfte.Ssl.TrustStore** credential. Note that the value of this property can contain environment variables.	No default value
wmqfte.Ssl.TrustStoreType	The type of SSL keystore you want to use. JKS and PKCS#12 keystores are supported. The value of this property can be either jks or pkcs12.	jks
wmqfte.Ssl.KeyStore	Specifies the location of the private key of the logger. The value of **wmqfte.Ssl.KeyStore** is a file path. Parentheses, commas (,) and backslashes (\) are special characters in MFT commands, and must be escaped with a backslash (\) character. Windows File paths on Windows can be specified either using double backslashes (\\) as a separator, or using single forward slashes (/). Note that the value of this property can contain environment variables.	No default value
wmqfte.Ssl.KeyStore.CredentialsFile	The path to the file that contains the **wmqfte.Ssl.KeyStore** credential. Note that the value of this property can contain environment variables.	No default value
wmqfte.Ssl.KeyStoreType	The type of SSL keystore you want to use. JKS and PKCS#12 keystores are supported. The value of this property can be either jks or pkcs12.	jks
wmqfte.Ssl.FipsRequired	Specifies that you want to enable FIPS support at the level of the logger. The value of this property can be true or false. For more information, see FIPS support in MFT.	false

Related concepts

SSL/TLS properties for MFT

Related reference

"The use of environment variables in MFT properties" on page 163

It is possible for environment variables to be used in Managed File Transfer properties that represent file or directory locations. This allows the locations of files or directories used when running parts of the product to vary depending on the current environment (such as the user running a command, for example).

"The MFT agent.properties file" on page 169

Each Managed File Transfer Agent has its own properties file, `agent.properties`, that must contain the information that an agent uses to connect to its queue manager. The `agent.properties` file can also contain properties that alter the behavior of the agent.

"The MFT command.properties file" on page 195

The `command.properties` file specifies the command queue manager to connect to when you issue commands and the information that Managed File Transfer requires to contact that queue manager.

"The MFT coordination.properties file" on page 191

The `coordination.properties` file specifies the connection details to the coordination queue manager. Because several Managed File Transfer installations might share the same coordination queue manager, you can use a symbolic link to a common `coordination.properties` file on a shared drive.

V 9.3.0 Output produced by the LogTransfer function

Transfer log events capture the details of transfer progress from the time transfer is submitted until it is completed. Information about transfer going into resynchronization is also captured to help you understand the progress of a transfer.

Transfer event format

Transfer events are in JSON format and written to the `transferlogN.json` file, which is created in the log directory of the agent, where `N` is a number with `0` being the default. Every event includes the following common attributes:

- Date and time (in UTC)
- Unique ID

There are additional attributes included in the event information written, depending on the type of the event and the level of transfer log. While the transfer log level *info* writes minimal information, the *verbose* level includes a more detailed information. The following "Sample events" on page 208 section describes a few examples of transfer events logged by an agent.

Unique ID

The unique id is included to help you easily identify the different phases as a transfer progresses, for example, `BFGTL0001`. The unique ID is part of the **`eventDescription`** attribute and is made up of two parts:

BFGTL
The prefix used for all identifiers, where BFG is the standard suffix used in Managed File Transfer and TL indicates this is a transfer log.

Number
A unique number starting from 1. For example:

```
{
  "eventDescription": "BFGTL0001: New transfer request submitted"
}
```

Sample events

The following table describes some of the events as examples of the information logged by the additional function. The second column of the table *Log Level* indicates the level at which the event is logged.

Important: The following attributes are included in the event information if the **`logTransfer`** level is set to *verbose* or *moderate*:

- **`sourceAgent`**
- **`destinationAgent`**
- **`threadId`**

Event	Log level	Description

List of items to transfer	verbose	<pre>{
		  "dateTime": "<Data time in UTC>",
		  "eventDescription": "BFGTL0002I: Generated detailed transfer item list.",
		  "destinationAgent": "<Name of destination agent>",
		  "sourceAgent": "<Name of source agent>",
		  "threadId": "0000001d",
		  "totalItemsInTransfer": <Number of items in the transfer>,
		  "transferId": "<Transfer Identifier>",
		  "transferItemsList": [{"source":"source item name",
		  "destination": "destination item name"}]
		}
		
		Example:
		{
		  "dateTime": "2022-01-14T12:56:54.219Z UTC",
		  "eventDescription": "BFGTL0002I: Generated detailed transfer item list.",
		  "destinationAgent": "QMBAGQ",
		  "sourceAgent": "QMBAG1",
		  "threadId": "0000001d",
		  "totalItems": 1,
		  "transferId": "414d5120514d412020202020202020202063bd17610a390040",
		  "transferItems": [{
		  "destination": "/results/rts/target/destFile.txt",
		  "source": "DESTINATIONQ@QMB"
		  }]
		}</pre>

Event	Log level	Description

Audit information of transfer logged by source agent	info, moderate, verbose	<pre>{
		  "dateTime": "<Date and time in UTC>",
		  "transferId": "<Transfer Id>",
		  "eventDescription": "BFGTL0008: Transfer progress information ",
		  "progressInformation": [{
		  "destination": "<Destination item name>",
		  "resultCode": "<Result of transfer>",
		  "supplement": "<Any supplemental information.>"
		  "source": "<Source item name>"
		  }],
		  "sourceAgent": "<Source agent name>",
		  "destinationAgent": "<Destination agent name>",
		  "threadId": "Thread Id",
		}
		
		Example:
		{
		  "dateTime": "2022-01-14T14:17:20.075Z UTC",
		  "transferId": "414D5120514D41202020202020202020202FD01761013B0040",
		  "eventDescription": "BFGTL0008: Transfer progress information",
		  "progressInformation": [{
		  "source": "MONITORQ@QMA"
		  "destination": "/results/rts/target/destinationFile.txt",
		  "resultCode": "Successful",
		  "supplement": ""
		  }],
		  "sourceAgent": "QMAAG1",
		  "destinationAgent": "QMBAG1",
		  "threadId": "00000023",
		}</pre>
		Following is an example of the information logged if the level is `info`:
		<pre>{
		  "dateTime":"2022-01-16T03:28:26.234Z UTC",
		"transferId":"414D51204D4654514D20202020202020EB89416101760040",
		"eventDescription":"BFGTL0008I: Transfer progress information",
		  "progressInformation": {
		  "warnings":0,
		  "failed":1,
		  "successful":0,
		  "unknown":0}
		}</pre>
		Following is an example of the information logged if the level is `moderate`:
		<pre>{
		"dateTime":"2022-01-16T03:30:46.766Z UTC",
		"transferId":"414D51204D4654514D20202020202020EB89416101850040",
		"eventDescription":"BFGTL0008I: Transfer progress information",
		"progressInformation": [{
		"source":"FileZilla:/home/mft/*",
		"destination":"C:/MFT/FileZilla:/home/mft/*",
		"itemSize":0,
		"resultCode":"Failed",
		"supplement":"BFGIO0110E: File \"/home/mft/*\" does not exist."}],
		"sourceAgent":"IMQFT01"
		"destinationAgent":"SRC",
		}</pre>

Error occurred when opening an item	info	<pre>{
		  "dateTime": "<Date and time in UTC>",
		  "transferId": "<Transfer Id>",
		  "eventDescription": "BFGTL0054E: Failed to open an item to transfer
		data.",
		  "itemName": "Name of file or queue",
		  "error":"<Error description>"
		  "sourceAgent": "<Source agent name>",
		  "destinationAgent": "<Destination agent name>",
		  "threadId": "<Thread Id>",
		}
		
		Example:
		{
		  "dateTime": "2022-01-14T14:42:24.902Z UTC",
		  "transferId": "414D512043514D485830312020202020B0D4176101370040",
		  "eventDescription": " BFGTL0054E: Failed to open an item to transfer
		data.",
		  "itemName":"/inbound/resource.xml",
		  "error": "File not found"
		  "sourceAgent": "CQMHX01AG1",
		  "destinationAgent": "HX01AG1",
		  "threadId": "0000001c"</pre>

Event	Log level	Description

List of transfers to be recovered at the start of the agent	verbose	<pre>{
		  "dateTime": "<Date and time in UTC>",
		  "eventDescription": "The list of transfers being recovered as part
		of agent recovery process.",
		  "agentName": "<Agent name>",
		  "transfers": [{"transferId":"<transfer state>"}]
		  "threadId": "<Thread Id>",
		}
		
		Example:
		{
		  "dateTime": "2022-01-14T14:42:24.902Z UTC",
		  "eventDescription": "The list of transfers being recovered as part
		of agent recovery process.",
		  "agentName": "CQMHX01AG1",
		  "transfers":
		[{"414D512043514D485830312020202020B0D4176101370040":"completeReceived
		"},
		{"414D512043514D485830312020202020B0D4176101370050":"resynchronizing"}]
		  "threadId": "0000001c",
		}</pre>

Related reference

“Java system properties for MFT” on page 211
A number of Managed File Transfer command and agent properties must be defined as Java system properties, because they define configuration for early function that is unable to use the command or agent properties mechanism.

fteCreateAgent

“The use of environment variables in MFT properties” on page 163
It is possible for environment variables to be used in Managed File Transfer properties that represent file or directory locations. This allows the locations of files or directories used when running parts of the product to vary depending on the current environment (such as the user running a command, for example).

Java system properties for MFT

A number of Managed File Transfer command and agent properties must be defined as Java system properties, because they define configuration for early function that is unable to use the command or agent properties mechanism.

Define system properties and other JVM options for the JVM that is to run Managed File Transfer commands by defining the environment variable BFG_JVM_PROPERTIES. For example, to set the com.ibm.wmqfte.maxConsoleLineLength property on a UNIX-type platform, define the variable as follows:

```
export BFG_JVM_PROPERTIES="-Dcom.ibm.wmqfte.maxConsoleLineLength=132"
```

If you are running an agent as a Windows service, you can modify the agent's Java system properties by specifying the -sj parameter on the **fteModifyAgent** command.

Table 113. Java system properties

Property name	Description	Value
com.ibm.wmqfte.maxConsoleLineLength	Maximum length of line that can be written to the console. Lines that exceed this length are word wrapped. This value is expressed in bytes (not characters).	IBM i The default length for IBM i is 132 bytes. z/OS ALW For z/OS, AIX, Linux, and Windows, the length is unlimited.

Table 113. Java system properties (continued)		
Property name	**Description**	**Value**
com.ibm.wmqfte.daemon.windows.windowsServiceLogFilesm	(Windows only.) Specifies the maximum number of Windows service log files to keep. Windows service log files are created in the agent and database logger logs directories if these applications are running as a Windows service. Windows service log files are named with the prefix *service*, and contain messages about the starting and stopping of the service.	5

Related concepts

MFT configuration options on Multiplatforms

Hints and tips for using MFT

SHA-2 CipherSpecs and CipherSuites for MFT

Managed File Transfer supports SHA-2 CipherSpecs and CipherSuites.

For more information about CipherSpecs and CipherSuites that are available for connections between agents and IBM MQ queue managers, see TLS CipherSpecs and CipherSuites in IBM MQ classes for Java and SSL/TLS CipherSpecs and CipherSuites in IBM MQ classes for JMS.

For more information about configuring CipherSpecs and CipherSuites for use with the protocol bridge agents (PBAs) and FTPS servers, see FTPS server support by the protocol bridge and Protocol bridge properties file format.

If you want to comply with SP 800-131A, you must satisfy the following requirements:

- You must use FTPS, which you have configured appropriately; SFTP is not supported.
- The remote server must send SP 800-131A-compliant cipher suites only.

Related concepts

SSL/TLS properties for MFT

MFT file logger configuration files

In addition to the `logger.properties` file, a Managed File Transfer stand-alone file logger also has an XML configuration file in its configuration directory. This configuration file is called `FileLoggerFormat.xml` and it defines the format used by the file logger to write messages to the log file. The content of this file must conform to the XML schema defined in the `FileLoggerFormat.xsd` file.

Related concepts

MFT stand-alone file logger format

Related reference

“The MFT logger.properties file” on page 199

The Managed File Transfer logger has a set of configuration properties. Specify these properties in the `logger.properties` file, which is in the *MQ_DATA_PATH*`/mqft/config/`*coordination_qmgr_name*`/loggers/`*logger_name* directory.

“MFT stand-alone file logger default log format” on page 213

Default log file format definition for the Managed File Transfer stand-alone file logger.

“Stand-alone file logger format XSD” on page 217

The schema for a stand-alone file format.

MFT stand-alone file logger default log format

Default log file format definition for the Managed File Transfer stand-alone file logger.

```
<?xml version="1.0" encoding="UTF-8"?>
<logFormatDefinition  xmlns:xsi="https://www.w3.org/2001/XMLSchema-instance"
  version="1.00" xsi:noNamespaceSchemaLocation="FileLoggerFormat.xsd">
  <messageTypes>
    <callCompleted>
      <format>
        <inserts>
          <insert type="user"   width="19" ignoreNull="false">/transaction/action/@time</insert>
          <insert type="user"   width="48" ignoreNull="false">/transaction/@ID</insert>
          <insert type="system" width="6"  ignoreNull="false">type</insert>
          <insert type="user"   width="3"  ignoreNull="false">/transaction/status/@resultCode</insert>
          <insert type="user"   width="0"  ignoreNull="false">/transaction/agent/@agent</insert>
          <insert type="user"   width="0"  ignoreNull="false">/transaction/agent/@QMgr</insert>
          <insert type="user"   width="0"  ignoreNull="false">/transaction/job/name</insert>
          <insert type="user"   width="0"  ignoreNull="true">/transaction/transferSet/call/command/
@type</insert>
          <insert type="user"   width="0"  ignoreNull="true">/transaction/transferSet/call/command/
@name</insert>
          <insert type="system" width="0"  ignoreNull="true">callArguments</insert>
          <insert type="user"   width="0"  ignoreNull="true">/transaction/transferSet/call/callResult/
@outcome</insert>
          <insert type="user"   width="0"  ignoreNull="true">/transaction/transferSet/call/callResult/
result/error</insert>
        </inserts>
        <separator>;</separator>
      </format>
    </callCompleted>
    <callStarted>
      <format>
        <inserts>
          <insert type="user"   width="19" ignoreNull="false">/transaction/action/@time</insert>
          <insert type="user"   width="48" ignoreNull="false">/transaction/@ID</insert>
          <insert type="system" width="6"  ignoreNull="false">type</insert>
          <insert type="user"   width="0"  ignoreNull="false">/transaction/agent/@agent</insert>
          <insert type="user"   width="0"  ignoreNull="false">/transaction/agent/@QMgr</insert>
          <insert type="user"   width="0"  ignoreNull="false">/transaction/job/name</insert>
          <insert type="user"   width="0"  ignoreNull="true">/transaction/transferSet/call/command/
@type</insert>
          <insert type="user"   width="0"  ignoreNull="true">/transaction/transferSet/call/command/
@name</insert>
          <insert type="system" width="0"  ignoreNull="true">callArguments</insert>
        </inserts>
        <separator>;</separator>
      </format>
    </callStarted>
    <monitorAction>
      <format>
        <inserts>
          <insert type="user"   width="19" ignoreNull="false">/monitorLog/action/@time</insert>
          <insert type="user"   width="48" ignoreNull="false">/monitorLog/@referenceId</insert>
          <insert type="system" width="6"  ignoreNull="false">type</insert>
          <insert type="user"   width="3"  ignoreNull="false">/monitorLog/status/@resultCode</insert>
          <insert type="user"   width="0"  ignoreNull="false">/monitorLog/@monitorName</insert>
          <insert type="user"   width="0"  ignoreNull="false">/monitorLog/monitorAgent/@agent</insert>
          <insert type="user"   width="0"  ignoreNull="false">/monitorLog/monitorAgent/@QMgr</insert>
          <insert type="user"   width="0"  ignoreNull="false">/monitorLog/action</insert>
        </inserts>
        <separator>;</separator>
      </format>
    </monitorAction>
    <monitorCreate>
      <format>
        <inserts>
          <insert type="user"   width="19" ignoreNull="false">/monitorLog/action/@time</insert>
          <insert type="user"   width="48" ignoreNull="false">/monitorLog/@referenceId</insert>
          <insert type="system" width="6"  ignoreNull="false">type</insert>
          <insert type="user"   width="0"  ignoreNull="false">/monitorLog/@monitorName</insert>
          <insert type="user"   width="0"  ignoreNull="false">/monitorLog/monitorAgent/@agent</insert>
          <insert type="user"   width="0"  ignoreNull="false">/monitorLog/monitorAgent/@QMgr</insert>
          <insert type="user"   width="0"  ignoreNull="false">/monitorLog/action</insert>
        </inserts>
        <separator>;</separator>
      </format>
```

```
    </monitorCreate>
    <monitorFired>
      <format>
        <inserts>
          <insert type="user"   width="19" ignoreNull="false">/monitorLog/action/@time</insert>
          <insert type="user"   width="48" ignoreNull="false">/monitorLog/@referenceId</insert>
          <insert type="system" width="6"  ignoreNull="false">type</insert>
          <insert type="user"   width="3"  ignoreNull="false">/monitorLog/status/@resultCode</insert>
          <insert type="user"   width="0"  ignoreNull="false">/monitorLog/@monitorName</insert>
          <insert type="user"   width="0"  ignoreNull="false">/monitorLog/monitorAgent/@agent</insert>
          <insert type="user"   width="0"  ignoreNull="false">/monitorLog/monitorAgent/@QMgr</insert>
          <insert type="user"   width="0"  ignoreNull="false">/monitorLog/action</insert>
          <insert type="user"   width="48" ignoreNull="false">/monitorLog/references/taskRequest</insert>
        </inserts>
        <separator>;</separator>
      </format>
    </monitorFired>
    <notAuthorized>
      <format>
        <inserts>
          <insert type="user"   width="19" ignoreNull="false">/notAuthorized/action/@time</insert>
          <insert type="user"   width="48" ignoreNull="false">/notAuthorized/@ID</insert>
          <insert type="system" width="6"  ignoreNull="false">type</insert>
          <insert type="user"   width="3"  ignoreNull="false">/notAuthorized/status/@resultCode</insert>
          <insert type="user"   width="12" ignoreNull="false">/notAuthorized/action</insert>
          <insert type="user"   width="12" ignoreNull="false">/notAuthorized/authority</insert>
          <insert type="user"   width="0"  ignoreNull="false">/notAuthorized/originator/userID</insert>
          <insert type="user"   width="0"  ignoreNull="false">/notAuthorized/status/supplement</insert>
        </inserts>
        <separator>;</separator>
      </format>
    </notAuthorized>
    <scheduleDelete>
      <format>
        <inserts>
          <insert type="user"   width="19" ignoreNull="false">/schedulelog/action/@time</insert>
          <insert type="user"   width="48" ignoreNull="false">/schedulelog/@ID</insert>
          <insert type="system" width="6"  ignoreNull="false">type</insert>
          <insert type="user"   width="3"  ignoreNull="false">/schedulelog/status/@resultCode</insert>
          <insert type="user"   width="0"  ignoreNull="false">/schedulelog/sourceAgent/@agent</insert>
          <insert type="user"   width="12" ignoreNull="false">/schedulelog/action</insert>
          <insert type="user"   width="0"  ignoreNull="false">/schedulelog/originator/userID</insert>
          <insert type="user"   width="0"  ignoreNull="true">/schedulelog/status/supplement</insert>
        </inserts>
        <separator>;</separator>
      </format>
    </scheduleDelete>
    <scheduleExpire>
      <format>
        <inserts>
          <insert type="user"   width="19" ignoreNull="false">/schedulelog/action/@time</insert>
          <insert type="user"   width="48" ignoreNull="false">/schedulelog/@ID</insert>
          <insert type="system" width="6"  ignoreNull="false">type</insert>
          <insert type="user"   width="3"  ignoreNull="false">/schedulelog/status/@resultCode</insert>
          <insert type="user"   width="0"  ignoreNull="false">/schedulelog/sourceAgent/@agent</insert>
          <insert type="user"   width="12" ignoreNull="false">/schedulelog/action</insert>
          <insert type="user"   width="0"  ignoreNull="false">/schedulelog/originator/userID</insert>
          <insert type="user"   width="0"  ignoreNull="true">/schedulelog/status/supplement</insert>
        </inserts>
        <separator>;</separator>
      </format>
    </scheduleExpire>
    <scheduleSkipped>
      <format>
        <inserts>
          <insert type="user"   width="19" ignoreNull="false">/schedulelog/action/@time</insert>
          <insert type="user"   width="48" ignoreNull="false">/schedulelog/@ID</insert>
          <insert type="system" width="6"  ignoreNull="false">type</insert>
          <insert type="user"   width="3"  ignoreNull="false">/schedulelog/status/@resultCode</insert>
          <insert type="user"   width="0"  ignoreNull="false">/schedulelog/sourceAgent/@agent</insert>
          <insert type="user"   width="12" ignoreNull="false">/schedulelog/action</insert>
          <insert type="user"   width="0"  ignoreNull="false">/schedulelog/originator/userID</insert>
          <insert type="user"   width="0"  ignoreNull="true">/schedulelog/status/supplement</insert>
        </inserts>
        <separator>;</separator>
      </format>
    </scheduleSkipped>
    <scheduleSubmitInfo>
      <format>
        <inserts>
          <insert type="user"   width="19" ignoreNull="false">/schedulelog/action/@time</insert>
          <insert type="user"   width="48" ignoreNull="false">/schedulelog/@ID</insert>
```

```
          <insert type="system" width="6"  ignoreNull="false">type</insert>
          <insert type="user"   width="3"  ignoreNull="false">/schedulelog/status/@resultCode</insert>
          <insert type="user"   width="0"  ignoreNull="false">/schedulelog/sourceAgent/@agent</insert>
          <insert type="user"   width="12" ignoreNull="false">/schedulelog/action</insert>
          <insert type="user"   width="0"  ignoreNull="false">/schedulelog/originator/userID</insert>
          <insert type="user"   width="0"  ignoreNull="true">/schedulelog/schedule/submit/</insert>
          <insert type="user"   width="0"  ignoreNull="true">/schedulelog/schedule/submit/@timezone</
insert>
          <insert type="user"   width="3"  ignoreNull="true">/schedulelog/schedule/repeat/frequency</
insert>
          <insert type="user"   width="12" ignoreNull="true">/schedulelog/schedule/repeat/frequency/
@interval</insert>
          <insert type="user"   width="3"  ignoreNull="true">/schedulelog/schedule/repeat/expireCount</
insert>
          <insert type="user"   width="0"  ignoreNull="true">/schedulelog/status/supplement</insert>
        </inserts>
        <separator>;</separator>
      </format>
    </scheduleSubmitInfo>
    <scheduleSubmitTransfer>
      <format>
        <inserts>
          <insert type="user"   width="19" ignoreNull="false">/schedulelog/action/@time</insert>
          <insert type="user"   width="48" ignoreNull="false">/schedulelog/@ID</insert>
          <insert type="system" width="10" ignoreNull="false">type</insert>
          <insert type="user"   width="0"  ignoreNull="false">/transaction/sourceAgent/@agent |
          /transaction/sourceWebUser/@webGatewayAgentName |
          /transaction/sourceWebGateway/@webGatewayAgentName</insert>
          <insert type="user"   width="0"  ignoreNull="false">/transaction/sourceAgent/@QMgr |
         /transaction/sourceWebUser/@webGatewayAgentQMgr |
        /transaction/sourceWebGateway/@webGatewayAgentQMgr</insert>
          <insert type="user"   width="0"  ignoreNull="false">/transaction/destinationAgent/@agent |
        /transaction/destinationWebUser/@webGatewayAgentName |
        /transaction/destinationWebGateway/@webGatewayAgentName</insert>
          <insert type="user"   width="0"  ignoreNull="false">/transaction/destinationAgent/@QMgr |
       /transaction/destinationWebUser/@webGatewayAgentQMgr |
       /transaction/destinationWebGateway/@webGatewayAgentQMgr</insert>
        </inserts>
        <separator>;</separator>
      </format>
    </scheduleSubmitTransfer>
    <scheduleSubmitTransferSet>
      <format>
        <inserts>
          <insert type="user"   width="19" ignoreNull="false">/schedulelog/action/@time</insert>
          <insert type="user"   width="48" ignoreNull="false">/schedulelog/@ID</insert>
          <insert type="system" width="10" ignoreNull="false">type</insert>
          <insert type="user"   width="0"  ignoreNull="false">source/file | source/queue</insert>
          <insert type="user"   width="5"  ignoreNull="true">source/@type</insert>
          <insert type="user"   width="6"  ignoreNull="true">source/@disposition</insert>
          <insert type="user"   width="0"  ignoreNull="false">destination/file | destination/queue</
insert>
          <insert type="user"   width="5"  ignoreNull="true">destination/@type</insert>
          <insert type="user"   width="9"  ignoreNull="true">destination/@exist</insert>
        </inserts>
        <separator>;</separator>
      </format>
    </scheduleSubmitTransferSet>
    <transferStarted>
      <format>
        <inserts>
          <insert type="user"   width="19" ignoreNull="false">/transaction/action/@time</insert>
          <insert type="user"   width="48" ignoreNull="false">/transaction/@ID</insert>
          <insert type="system" width="6"  ignoreNull="false">type</insert>
          <insert type="user"   width="3"  ignoreNull="true">/transaction/status/@resultCode</insert>
          <insert type="user"   width="0"  ignoreNull="false">/transaction/sourceAgent/@agent |
        /transaction/sourceWebUser/@webGatewayAgentName |
        /transaction/sourceWebGateway/@webGatewayAgentName</insert>
          <insert type="user"   width="0"  ignoreNull="true">/transaction/sourceAgent/@QMgr |
       /transaction/sourceWebUser/@webGatewayAgentQMgr |
       /transaction/sourceWebGateway/@webGatewayAgentQMgr</insert>
          <insert type="user"   width="0"  ignoreNull="true">/transaction/sourceAgent/@agentType |
       /transaction/sourceWebUser/@webGatewayAgentType |
       /transaction/sourceWebGateway/@webGatewayAgentType</insert>
          <insert type="user"   width="0"  ignoreNull="false">/transaction/destinationAgent/@agent |
       /transaction/destinationWebUser/@webGatewayAgentName |
       /transaction/destinationWebGateway/@webGatewayAgentName</insert>
          <insert type="user"   width="0"  ignoreNull="true">/transaction/destinationAgent/@QMgr |
       /transaction/destinationWebUser/@webGatewayAgentQMgr |
       /transaction/destinationWebGateway/@webGatewayAgentQMgr</insert>
          <insert type="user"   width="0"  ignoreNull="true">/transaction/originator/userID</insert>
          <insert type="user"   width="0"  ignoreNull="true">/transaction/job/name</insert>
```

```
        <insert type="user"   width="0"  ignoreNull="true">/transaction/scheduleLog/@ID</insert>
      </inserts>
      <separator>;</separator>
    </format>
  </transferStarted>
  <transferCancelled>
    <format>
      <inserts>
        <insert type="user"   width="19" ignoreNull="false">/transaction/action/@time</insert>
        <insert type="user"   width="48" ignoreNull="false">/transaction/@ID</insert>
        <insert type="system" width="6"  ignoreNull="false">type</insert>
        <insert type="user"   width="3"  ignoreNull="true">/transaction/status/@resultCode</insert>
        <insert type="user"   width="0"  ignoreNull="false">/transaction/sourceAgent/@agent |
    /transaction/sourceWebUser/@webGatewayAgentName |
    /transaction/sourceWebGateway/@webGatewayAgentName</insert>
        <insert type="user"   width="0"  ignoreNull="true">/transaction/sourceAgent/@QMgr |
    /transaction/sourceWebUser/@webGatewayAgentQMgr |
    /transaction/sourceWebGateway/@webGatewayAgentQMgr</insert>
        <insert type="user"   width="0"  ignoreNull="true">/transaction/sourceAgent/@agentType |
    /transaction/sourceWebUser/@webGatewayAgentType |
    /transaction/sourceWebGateway/@webGatewayAgentType</insert>
        <insert type="user"   width="0"  ignoreNull="false">/transaction/destinationAgent/@agent |
   /transaction/destinationWebUser/@webGatewayAgentName |
   /transaction/destinationWebGateway/@webGatewayAgentName</insert>
        <insert type="user"   width="0"  ignoreNull="true">/transaction/destinationAgent/@QMgr |
   /transaction/destinationWebUser/@webGatewayAgentQMgr |
   /transaction/destinationWebGateway/@webGatewayAgentQMgr</insert>
        <insert type="user"   width="0"  ignoreNull="true">/transaction/destinationAgent/@agentType |
  /transaction/destinationWebUser/@webGatewayAgentType |
  /transaction/destinationWebGateway/@webGatewayAgentType</insert>
        <insert type="user"   width="0"  ignoreNull="true">/transaction/originator/userID</insert>
        <insert type="user"   width="0"  ignoreNull="true">/transaction/job/name</insert>
        <insert type="user"   width="0"  ignoreNull="true">/transaction/status/supplement</insert>
      </inserts>
      <separator>;</separator>
    </format>
  </transferCancelled>
  <transferComplete>
    <format>
      <inserts>
        <insert type="user"   width="19" ignoreNull="false">/transaction/action/@time</insert>
        <insert type="user"   width="48" ignoreNull="false">/transaction/@ID</insert>
        <insert type="system" width="6"  ignoreNull="false">type</insert>
        <insert type="user"   width="3"  ignoreNull="true">/transaction/status/@resultCode</insert>
        <insert type="user"   width="0"  ignoreNull="false">/transaction/sourceAgent/@agent |
         /transaction/sourceWebUser/@webGatewayAgentName |
        /transaction/sourceWebGateway/@webGatewayAgentName</insert>
        <insert type="user"   width="0"  ignoreNull="true">/transaction/sourceAgent/@QMgr |
        /transaction/sourceWebUser/@webGatewayAgentQMgr |
        /transaction/sourceWebGateway/@webGatewayAgentQMgr</insert>
        <insert type="user"   width="0"  ignoreNull="true">/transaction/sourceAgent/@agentType |
        /transaction/sourceWebUser/@webGatewayAgentType |
        /transaction/sourceWebGateway/@webGatewayAgentType</insert>
        <insert type="user"   width="0"  ignoreNull="false">/transaction/destinationAgent/@agent |
        /transaction/destinationWebUser/@webGatewayAgentName |
        /transaction/destinationWebGateway/@webGatewayAgentName</insert>
        <insert type="user"   width="0"  ignoreNull="true">/transaction/destinationAgent/@QMgr |
        /transaction/destinationWebUser/@webGatewayAgentQMgr |
        /transaction/destinationWebGateway/@webGatewayAgentQMgr</insert>
        <insert type="user"   width="0"  ignoreNull="true">/transaction/destinationAgent/@agentType |
        /transaction/destinationWebUser/@webGatewayAgentType |
        /transaction/destinationWebGateway/@webGatewayAgentType</insert>
        <insert type="user"   width="0"  ignoreNull="true">/transaction/originator/userID</insert>
        <insert type="user"   width="0"  ignoreNull="true">/transaction/job/name</insert>
        <insert type="user"   width="0"  ignoreNull="true">/transaction/status/supplement</insert>
      </inserts>
      <separator>;</separator>
    </format>
  </transferComplete>
  <transferDelete>
    <format>
      <inserts>
        <insert type="user"   width="19" ignoreNull="false">/transaction/action/@time</insert>
        <insert type="user"   width="48" ignoreNull="false">/transaction/@ID</insert>
        <insert type="system" width="6"  ignoreNull="false">type</insert>
        <insert type="user"   width="3"  ignoreNull="true">/transaction/status/@resultCode</insert>
        <insert type="user"   width="0"  ignoreNull="false">/transaction/sourceAgent/@agent |
        /transaction/sourceWebUser/@webGatewayAgentName |
        /transaction/sourceWebGateway/@webGatewayAgentName</insert>
        <insert type="user"   width="0"  ignoreNull="true">/transaction/sourceAgent/@QMgr |
        /transaction/sourceWebUser/@webGatewayAgentQMgr |
        /transaction/sourceWebGateway/@webGatewayAgentQMgr</insert>
```

```
          <insert type="user"   width="0"  ignoreNull="true">/transaction/sourceAgent/@agentType |
          /transaction/sourceWebUser/@webGatewayAgentType |
          /transaction/sourceWebGateway/@webGatewayAgentType</insert>
          <insert type="user"   width="0"  ignoreNull="false">/transaction/destinationAgent/@agent |
          /transaction/destinationWebUser/@webGatewayAgentName |
          /transaction/destinationWebGateway/@webGatewayAgentName</insert>
          <insert type="user"   width="0"  ignoreNull="true">/transaction/destinationAgent/@QMgr |
          /transaction/destinationWebUser/@webGatewayAgentQMgr |
          /transaction/destinationWebGateway/@webGatewayAgentQMgr</insert>
          <insert type="user"   width="0"  ignoreNull="true">/transaction/destinationAgent/@agentType |
          /transaction/destinationWebUser/@webGatewayAgentType |
          /transaction/destinationWebGateway/@webGatewayAgentType</insert>
          <insert type="user"   width="0"  ignoreNull="true">/transaction/originator/userID</insert>
          <insert type="user"   width="0"  ignoreNull="true">/transaction/job/name</insert>
          <insert type="user"   width="0"  ignoreNull="true">/transaction/status/supplement</insert>
        </inserts>
        <separator>;</separator>
      </format>
    </transferDelete>
    <transferProgress>
      <format>
        <inserts>
          <insert type="user"   width="19" ignoreNull="false">/transaction/action/@time</insert>
          <insert type="user"   width="48" ignoreNull="false">/transaction/@ID</insert>
          <insert type="system" width="6"  ignoreNull="false">type</insert>
          <insert type="user"   width="3"  ignoreNull="true">status/@resultCode</insert>
          <insert type="user"   width="0"  ignoreNull="false">source/file | source/queue</insert>
          <insert type="user"   width="0"  ignoreNull="false">source/file/@size | source/queue/@size</
insert>
          <insert type="user"   width="5"  ignoreNull="true">source/@type</insert>
          <insert type="user"   width="6"  ignoreNull="true">source/@disposition</insert>
          <insert type="user"   width="0"  ignoreNull="true">source/file/@alias | source/queue/@alias</
insert>
          <insert type="user"   width="0"  ignoreNull="true">source/file/@filespace | source/queue/
@filespace</insert>
          <insert type="user"   width="0"  ignoreNull="true">source/@correlationBoolean1</insert>
          <insert type="user"   width="0"  ignoreNull="true">source/@correlationNum1</insert>
          <insert type="user"   width="0"  ignoreNull="true">source/@correlationString1</insert>
          <insert type="user"   width="0"  ignoreNull="false">destination/file | destination/queue</
insert>
          <insert type="user"   width="0"  ignoreNull="false">destination/file/@size | destination/queue/
@size</insert>
          <insert type="user"   width="5"  ignoreNull="true">destination/@type</insert>
          <insert type="user"   width="9"  ignoreNull="true">destination/@exist</insert>
          <insert type="user"   width="0"  ignoreNull="true">destination/file/@alias | destination/queue/
@alias</insert>
          <insert type="user"   width="0"  ignoreNull="true">destination/file/@filespace | destination/
queue/@filespace</insert>
          <insert type="user"   width="0"  ignoreNull="true">destination/file/@truncateRecords</insert>
          <insert type="user"   width="0"  ignoreNull="true">destination/@correlationBoolean1</insert>
          <insert type="user"   width="0"  ignoreNull="true">destination/@correlationNum1</insert>
          <insert type="user"   width="0"  ignoreNull="true">destination/@correlationString1</insert>
          <insert type="user"   width="0"  ignoreNull="true">status/supplement</insert>
        </inserts>
        <separator>;</separator>
      </format>
    </transferProgress>
  </messageTypes>
</logFormatDefinition>
```

Related reference

MFT stand-alone file logger format

“Stand-alone file logger format XSD” on page 217

The schema for a stand-alone file format.

Stand-alone file logger format XSD

The schema for a stand-alone file format.

Schema

```
<?xml version="1.0" encoding="UTF-8"?>
<!--
@start_non_restricted_prolog@
Version: %Z% %I% %W% %E% %U% [%H% %T%]
```

```
Licensed Materials - Property of IBM

5724-H72

Copyright IBM Corp. 2011, 2024. All Rights Reserved.

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

<!--
     This schema defines the format of the FileLoggerFormat XML file that contains the definition
     of the format to use when logging FTE log messages to a file. When an XML file that conforms
     to this schema is processed by a file logger it can contain definitions for one or more
     message type(s) that define how log messages of those types are output to the file log.
-->

<xsd:schema xmlns:xsd="https://www.w3.org/2001/XMLSchema">

<xsd:include schemaLocation="fteutils.xsd"/>

    <!--
        Defines the logFileDefinition and version number
        <logFileDefinition version="1.00" ...
            <messageTypes>
                ...
            </messageTypes>
        </logFileDefinition>
     -->
    <xsd:element name="logFileDefinition">
        <xsd:complexType>
            <xsd:sequence>
                <xsd:element   name="messageTypes" type="messageTypesType" maxOccurs="1" minOccurs="1"/>
            </xsd:sequence>
            <xsd:attribute name="version"      type="versionType"      use="required"/>
        </xsd:complexType>
    </xsd:element>

    <!--
        Defines the set of accepted message types. The definition of individual message types
        is optional. If a particular types element is present but empty then no line will be
        output for messages of that type. If a particular types element is not present then
        the default format will be used to format messages of that type.
     -->
    <xsd:complexType name="messageTypesType">
         <xsd:sequence>
            <xsd:element name="callCompleted"             type="messageType"  maxOccurs="1"
minOccurs="0"/>
            <xsd:element name="callStarted"               type="messageType"  maxOccurs="1"
minOccurs="0"/>
            <xsd:element name="monitorAction"             type="messageType"  maxOccurs="1"
minOccurs="0"/>
            <xsd:element name="monitorCreate"             type="messageType"  maxOccurs="1"
minOccurs="0"/>
            <xsd:element name="monitorFired"              type="messageType"  maxOccurs="1"
minOccurs="0"/>
            <xsd:element name="notAuthorized"             type="messageType"  maxOccurs="1"
minOccurs="0"/>
            <xsd:element name="scheduleDelete"            type="messageType"  maxOccurs="1"
minOccurs="0"/>
            <xsd:element name="scheduleExpire"            type="messageType"  maxOccurs="1"
minOccurs="0"/>
            <xsd:element name="scheduleSkipped"           type="messageType"  maxOccurs="1"
minOccurs="0"/>
            <xsd:element name="scheduleSubmitInfo"        type="messageType"  maxOccurs="1"
minOccurs="0"/>
            <xsd:element name="scheduleSubmitTransfer"    type="messageType"  maxOccurs="1"
minOccurs="0"/>
            <xsd:element name="scheduleSubmitTransferSet" type="messageType"  maxOccurs="1"
minOccurs="0"/>
            <xsd:element name="transferStarted"           type="messageType"  maxOccurs="1"
minOccurs="0"/>
            <xsd:element name="transferCancelled"         type="messageType"  maxOccurs="1"
minOccurs="0"/>
            <xsd:element name="transferComplete"          type="messageType"  maxOccurs="1"
minOccurs="0"/>
            <xsd:element name="transferDelete"            type="messageType"  maxOccurs="1"
minOccurs="0"/>
            <xsd:element name="transferProgress"          type="messageType"  maxOccurs="1"
minOccurs="0"/>
```

```
        </xsd:sequence>
    </xsd:complexType>

    <!--
        Defines the content of a message type definition e.g.

        <callStarted>
         <format>
          ...
         </format>
        <callStarted>
     -->
    <xsd:complexType name="messageType">
        <xsd:sequence>
            <xsd:element name="format"   type="messageFormatType"  maxOccurs="1" minOccurs="0"/>
        </xsd:sequence>
    </xsd:complexType>

    <!--
        Defines the content of a message format definition e.g.

         <format>
          <inserts>
           ...
          </inserts
          <separator>;</separator>
         </format>
     -->
    <xsd:complexType name="messageFormatType">
        <xsd:sequence>
            <xsd:element name="inserts"   type="insertsType"  maxOccurs="1" minOccurs="1"/>
            <xsd:element name="separator" type="scheduleType" maxOccurs="1" minOccurs="1"/>
        </xsd:sequence>
    </xsd:complexType>

    <!--
        Defines the content of the inserts element e.g.

        <inserts>
         <insert ...>
         <insert ...>
         ...
        </inserts>
     -->
    <xsd:complexType name="insertsType">
        <xsd:sequence>
             <xsd:element name="insert" type="insertType" maxOccurs="unbounded" minOccurs="1"/>
        </xsd:sequence>
    </xsd:complexType>

    <!--
        Defines the content of an insert definition e.g.

        <insert type="user" width="0" ignoreNull="true">/transaction/@ID</insert>
     -->
    <xsd:complexType name="insertType">
        <xsd:attribute name="type"       type="insertTypeType"         use="required"/>
        <xsd:attribute name="width"      type="xsd:nonNegativeInteger" use="required"/>
        <xsd:attribute name="ignoreNull" type="xsd:boolean"            use="required"/>
    </xsd:complexType>

    <!--
        Defines the accepted choices for the insert type attribute.
     -->
    <xsd:simpleType name="insertTypeType">
        <xsd:restriction base="xsd:token">
            <xsd:enumeration value="user"/>
            <xsd:enumeration value="system"/>
        </xsd:restriction>
    </xsd:simpleType>

</xsd:schema>
```

Related reference

MFT stand-alone file logger format

“MFT stand-alone file logger default log format” on page 213

Default log file format definition for the Managed File Transfer stand-alone file logger.

The SCSQFCMD library

The SCSQFCMD library provided by IBM MQ Managed File Transfer for z/OS contains members that act as templates for jobs which can be used to create a Managed File Transfer configuration, and create and administer an agent or logger.

The contents of the library are shown in the following table.

Member	Description
BFGCOPY	Job used to create a copy of the SCSQFCMD library
BFGCUSTM	Job used to customize a copy of the library for an agent or logger
BFGXCROB	**fteObfuscate** sample template.
BFGXLGCR	**fteCreateLogger** template.
BFGXMNCR	**fteCreateMonitor** sample template.
BFGXMNDE	**fteDeleteMonitor** sample template.
BFGXPRAN	**fteAnt** sample template
BFGXSTDE	**fteDeleteScheduledTransfer** sample template
BFGXTMCR	**fteCreateTemplate** sample template
BFGXTMDE	**fteDeleteTemplate** sample template
BFGXTRCA	**fteCancelTransfer** sample template
BFGXTRCR	**fteCreateTransfer** sample template
BFGYAGST	Template for a started task procedure to start an agent
BFGYLGST	Template for a started task procedure to start a logger
BFGZAGCL	**fteCleanAgent** sample template
BFGZAGCR	**fteCreateAgent** sample template
BFGZAGDE	**fteDeleteAgent** sample template
BFGZAGLG	**fteSetAgentLogLevel** sample template
BFGZAGLI	**fteListAgents** sample template
BFGZAGPI	**ftePingAgent** sample template
BFGZAGSH	**fteShowAgentDetails** sample template
BFGZAGSP	**fteStopAgent** sample template
BFGZAGST	**fteStartAgent** sample template
BFGZAGTC	**fteSetAgentTraceLevel** sample template
BFGZCFCR	**fteSetupCoordination** sample template
BFGZCFDF	**fteChangeDefaultConfigurationOptions** sample template
BFGZCMCR	**fteSetupCommands** sample template
BFGZCMD	Template for REXX script used by other members in the data set
BFGZLGDE	**fteDeleteLogger** sample template

Member	Description
BFGZLGSH	**fteShowLoggerDetails** sample template
BFGZLGSP	**fteStopLogger** sample template
BFGZLGST	**fteStartLogger** sample template
BFGZLGTC	**fteSetLoggerTraceLevel** sample template
BFGZMNLI	**fteListMonitors** sample template
BFGZPID	**fteSetProductId** sample template
BFGZPROF	Template for shell script used by other members in the data set
BFGZPRSH	**fteDisplayVersion** sample template
BFGZRAS	**fteRas** sample template
BFGZSTLI	**fteListScheduledTransfers** sample template
BFGZTMLI	**fteListTemplates** sample template

For details about how the SCSQFCMD library is used to generate a new library for creating a Managed File Transfer configuration, and creating and administering an agent or logger, see Configuring Managed File Transfer for z/OS.

Related reference

"The use of environment variables in MFT properties" on page 163
It is possible for environment variables to be used in Managed File Transfer properties that represent file or directory locations. This allows the locations of files or directories used when running parts of the product to vary depending on the current environment (such as the user running a command, for example).

SYSTEM.FTE topic

The SYSTEM.FTE topic is a topic on the coordination queue manager that Managed File Transfer uses to log transfers and store information about agents, monitors, schedules, and templates.

Topic structure

```
SYSTEM.FTE
      /Agents
            /agent_name
      /monitors
            /agent_name
      /Scheduler
            /agent_name
      /Templates
            /template_ID
      /Transfers
            /agent_name
                  /transfer_ID
      /Log
            /agent_name
                  /Monitors
                  /schedule_ID
                  /transfer_ID
```

SYSTEM.FTE/Agents/*agent_name*

This topic contains a retained publication that describes an agent in your Managed File Transfer network and its properties. The message on this topic is updated periodically with the agent status. For more information, see MFT agent status message format.

SYSTEM.FTE/monitors/*agent_name*
This topic contains retained publications that describe the resource monitors associated with the agent *agent_name*. The XML of the retained publication conforms to the schema `MonitorList.xsd`. For more information, see MFT monitor list message format.

SYSTEM.FTE/Scheduler/*agent_name*
This topic contains a retained publication that describes all of the active schedules that are associated with the agent *agent_name*. The XML of the retained publication conforms to the schema `ScheduleList.xsd`. For more information, see MFT schedule list message format.

SYSTEM.FTE/Templates
This topic contains retained publications that describe all of the templates that are defined in your Managed File Transfer topology.

- The publication that is associated with each template is published to a subtopic with the name SYSTEM.FTE/Templates/*template_ID*.

For an example of the contents of this retained publication, see MFT example template XML message.

SYSTEM.FTE/Transfers/*agent_name*
This topic contains publications that describe that status of transfers that originate at the agent *agent_name*. The publications that are associated with each transfer are published to a subtopic with the name SYSTEM.FTE/Transfers/*agent_name*/*transfer_ID*. These publications are used by the IBM MQ Explorer plug-in to provide progress information about individual transfers. The XML of the publication conforms to the schema `TransferStatus.xsd`. For more information, see File transfer status message format.

SYSTEM.FTE/Log/*agent_name*
This topic contains publications that log information about transfers, monitors, and schedules that originate at the agent *agent_name*. These publications can be logged by the database logger to provide audit records of the events that happen in your Managed File Transfer network.

- The publications that are associated with each transfer are published to a subtopic with the name SYSTEM.FTE/Log/*agent_name*/*transfer_ID* and the XML of the publication conforms to the schema `TransferLog.xsd`. For more information, see File transfer log message formats.
- The publications that are associated with each scheduled transfer are published to a subtopic with the name SYSTEM.FTE/Log/*agent_name*/*schedule_ID* and the XML of the publication conforms to the schema `ScheduleLog.xsd`. For more information, see Scheduled file transfer log message formats.
- The publications that are associated with each monitor are published to a subtopic with the name SYSTEM.FTE/Log/*agent_name*/*Monitors*/*monitor_name*/*monitor_ID* and the XML of the publication conforms to the schema `MonitorLog.xsd`. For more information, see MFT monitor log message format.

MFT Agent queue settings

The MQSC command scripts generated by the **fteCreateAgent** command create the agent queues with parameters set to the following values. If you do not use the MQSC scripts provided to create the queues, but create the queues manually, ensure you set the following parameters to the values given.

Agent operation queues

The agent's operation queues have the following names:

- SYSTEM.FTE.COMMAND.*agent_name*
- SYSTEM.FTE.DATA.*agent_name*
- SYSTEM.FTE.EVENT.*agent_name*
- SYSTEM.FTE.REPLY.*agent_name*
- SYSTEM.FTE.STATE.*agent_name*

Table 114. Agent operation queue parameters	
Parameter	**Value (if applicable)**
DEFPRTY	0
DEFSOPT	SHARED
GET	ENABLED
MAXDEPTH	5000
MAXMSGL	4194304
MSGDLVSQ	PRIORITY
PUT	ENABLED
RETINTVL	999999999
SHARE	
NOTRIGGER	
USAGE	NORMAL
REPLACE	

Agent authority queues

The agent's authority queues have the following names:

- SYSTEM.FTE.AUTHADM1.*agent_name*
- SYSTEM.FTE.AUTHAGT1. *agent_name*
- SYSTEM.FTE.AUTHMON1.*agent_name*
- SYSTEM.FTE.AUTHOPS1.*agent_name*
- SYSTEM.FTE.AUTHSCH1.*agent_name*
- SYSTEM.FTE.AUTHTRN1.*agent_name*

Table 115. Agent authority queue parameters	
Parameter	**Value (if applicable)**
DEFPRTY	0
DEFSOPT	SHARED
GET	ENABLED
MAXDEPTH	0
MAXMSGL	0
MSGDLVSQ	PRIORITY
PUT	ENABLED
RETINTVL	999999999
SHARE	
NOTRIGGER	
USAGE	NORMAL
REPLACE	

Related reference
fteCreateAgent (create an MFT agent)

MFT system queues and the system topic

Managed File Transfer has a number of system queues and one system topic that are for internal use only.

Any queues with a name beginning SYSTEM.FTE are internal system queues for Managed File Transfer (MFT). Do not delete these queues, as doing so prevents IBM MQ MFT from working correctly. Table 116 on page 224 shows which type of message is on each queue:

Table 116. Queue names, type and usage

Queue name	Queue type	Usage
SYSTEM.FTE.AUTHAGT1.agent_name	Authority	Queue for configuring authority for sending and receiving transfer requests.
SYSTEM.FTE.AUTHTRN1.agent_name	Authority	Queue for configuring authority to start and cancel managed transfers. Also to start managed calls.
SYSTEM.FTE.AUTHMON1.agent_name	Authority	Queue for configuring authority to allow a user to create or delete resource monitors that were created by the same user.
SYSTEM.FTE.AUTHOPS1.agent_name	Authority	Queue for configuring authority to delete resource monitors and scheduled transfers that were created by another user.
SYSTEM.FTE.AUTHSCH1.agent_name	Authority	Queue for configuring authority to create or delete scheduled transfers that were created by the same user.
SYSTEM.FTE.AUTHADM1.agent_name	Authority	Queue for configuring authority to shut down the agent, using the **-m** option on the **fteStopAgent** command.
SYSTEM.FTE.COMMAND.agent_name	Operation	Queue for sending command requests to an agent.
SYSTEM.FTE.DATA.agent_name	Operation	Queue used by a destination agent for holding data sent by a source agent.
SYSTEM.FTE.REPLY.agent_name	Operation	Queue for receiving replies from a destination agent.
SYSTEM.FTE.STATE.agent_name	Operation	Queue for holding the status of a transfer request.
SYSTEM.FTE.EVENT.agent_name	Operation	Queue for holding resource monitor history.
SYSTEM.FTE.HA.agent_name	Operation	Queue used as a lock by highly available agent instances.

If an agent is participating in message-to-file or file-to-message transfers, the definition of the SYSTEM.FTE.STATE.agent_name queue might need to be modified to allow these managed transfers to take place. For more information on this, see Guidance for setting MQ attributes and MFT properties associated with message size.

Attention: You must not change the definitions of the other system queues.

Also, do not modify or delete the SYSTEM.FTE topic as this is also for internal use only.

Temporary queues

Managed File Transfer creates temporary queues for a number of purposes. The name of each queue starts with WMQFTE. by default. (The period is part of the default prefix.) If you want to change this prefix, you can use the **dynamicQueuePrefix** property in the `command.properties` file or the `coordination.properties` file or both. The property in the `command.properties` file is used to set the prefix of temporary queues that are created for responses to commands that require a response from the agent. The property in the `coordination.properties` file is used to set the prefix of temporary queues that are created for other purposes; for example, the *WMQFTE.*FTE.TIMECHCK.QUEUE, where *WMQFTE.* is the value defined by the **dynamicQueuePrefix** property.

Related reference

Restricting user authorities on MFT agent actions

MFT object naming conventions

Use the following naming conventions for your Managed File Transfer objects:

- Agent and logger names:
  - Can be a maximum of 28 characters long and are not case-sensitive.
  - Entered in lowercase or mixed case are converted to uppercase
  - Must conform to standard IBM MQ object naming conventions.

    These conventions are detailed as follows: Rules for naming IBM MQ objects.
- In addition to the IBM MQ object naming conventions, the:
  - Forward slash (/) character cannot be used in agent names or logger names
  - Percent (%) character cannot be used in agent names or logger names.
- The names of properties in the properties files are case-sensitive.
- Queue manager names are case-sensitive.
- File names are case-sensitive for some platforms.
- Resource monitor and transfer template names:
  - Are not case-sensitive
  - Entered in lowercase or mixed case are converted to uppercase
  - Must not contain asterisk (*), percent (%) or question mark (?) characters
- Protocol file server names must:
  - Be a minimum of 2 characters long, but there is no maximum length limit
  - Are not case-sensitive
  - Must conform to standard IBM MQ object naming conventions.

    These conventions are detailed as follows: Rules for naming IBM MQ objects.

Files in the IBM i integrated file system (IFS)

IBM i File names in the IFS cannot contain any of the following characters:

- Backslash (\)

- Forward slash (/)
- Colon (:)
- Asterisk (*)
- Question mark (?)
- Quotation marks (")
- Less than symbol (<)
- Greater than symbol (>)
- Vertical bar (|)

If you attempt to transfer files with names containing any of these characters to an IBM i IFS, the transfer of these files fails.

Data set names

z/OS Data sets have naming restrictions, which affect the maximum name length and the available characters that you can use for data set names. PDS data set member names can be a maximum of eight characters and cannot contain the dot (.) character. When you transfer to a data set, you must explicitly specify the name, which means these naming restrictions do not cause a problem. But when you transfer from files to PDS members the file path might not map to a PDS member name. When you transfer to a PDS data set, each source file becomes a PDS member and each member name is generated from the name of the source.

PDS member names are z/OS unqualified names and are defined by the following regular expression:

```
[a-zA-Z$#@][a-zA-Z0-9$#@]{0-7}
```

The following scheme is used to convert a source data set or source file name to a valid PDS member name. The considerations are applied in the order listed:

1. Only the characters in the name after the last forward slash (/), the last backslash (\), or the last colon (:) character, are used. That is, only the name part of a file path is used.
2. For source files (not data sets or PDS members), the characters after and including the last dot (.) character, are ignored.
3. For any name longer than eight characters, the last eight characters only are used.
4. Dot characters are replaced with at sign (@) characters.
5. Invalid characters are replaced with at sign (@) characters.
6. If the conversion produces no characters, the PDS member name is `@`.

MFT agent status messages

High availability agents publish status information in XML format.

Sample XML showing information about three standby instances

```
<?xml version="1.0" encoding="UTF-8"?>
<AgentStandbyStatus version="6.00" xmlns:xsi="https://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="AgentStandbyStatus.xsd">
    <instance host="9.122.123.124" agentVersion="9.1.4.0" />
    <instance host="agenthost.ibm.com" agentVersion="9.1.4.0" />
    <instance host="10.11.12.14" agentVersion="9.1.4.0" />
</AgentStandby>
```

Agent status publication with standby status XML embedded.

The standby status XML is shown in bold type.

```
<?xml version="1.0" encoding="UTF-8"?>
<properties version="1.0">
  <entry key="SourceTransferStates"/>
  <entry key="queueManagerPort">1414</entry>
  <entry key="agentStandbyInstances">&lt;?xml version="1.0" encoding="UTF-8"?&gt;&lt;AgentStandbyStatus
version="6.00"
        xmlns:xsi="https://www.w3.org/2001/XMLSchema-instance"
        xsi:noNamespaceSchemaLocation="AgentStandbyStatus.xsd"&gt;&lt;Instances&gt;&lt;instance
host="9.122.123.124"
        agentVersion="9.1.4.0" /&gt;&lt;instance host="agenthost.ibm.com" agentVersion="9.1.4.0" /
&gt;&lt;instance host="10.11.12.14"
        agentVersion="9.1.4.0" /&gt;&lt;/Instances&gt;&lt;/AgentStandbyStatus&gt;</entry>
  <entry key="agentType">STANDARD</entry>
  <entry key="agentDeclaredHostName">MFTHA1</entry>
  <entry key="agentDescription"/>
  <entry key="maxQueuedTransfers">1000</entry>
  <entry key="agentTimeZone">America/Los_Angeles</entry>
  <entry key="agentOsName">Windows Server 2012 R2</entry>
  <entry key="PublishTimeUTC">2019-05-22T06:02:50Z</entry>
  <entry key="queueManagerHost">localhost</entry>
  <entry key="AgentStartTimeUTC">2019-05-22T04:13:02Z</entry>
  <entry key="agentTraceLevel">&lt;?xml version="1.0" encoding="UTF-8"?&gt;&lt;
        agentTraceStatus version="6.00" xmlns:xsi="https://www.w3.org/2001/XMLSchema-instance"
        xsi:noNamespaceSchemaLocation="AgentTraceStatus.xsd"&gt;&lt;trace
        level="all"&gt;com.ibm.wmqfte&lt;/trace&gt;&lt;/agentTraceStatus&gt;</entry>
  <entry key="DestinationTransferStates"/>
  <entry key="queueManager">MFTHAQM</entry>
  <entry key="agentProductVersion">9.1.4.0</entry>
  <entry key="AgentStatusPublishRate">300</entry>
  <entry key="maxSourceTransfers">25</entry>
  <entry key="AgentStatus">STARTED</entry>
  <entry key="maxDestinationTransfers">25</entry>
  <entry key="agentName">SRC</entry>
  <entry key="CommandTimeUTC">2019-05-22T06:02:50Z</entry>
  <entry key="queueManagerChannel">MFT_HA_CHN</entry>
  <entry key="agentInterfaceVersion">6.00</entry>
  <entry key="agentVersion">p914-L191119</entry>
</properties>
```

Related reference

fteCreateAgent

agent GET

IBM MQ Internet Pass-Thru configuration reference

IBM MQ Internet Pass-Thru (MQIPT) uses a configuration file called `mqipt.conf` to define routes and to control the actions of the MQIPT server. From IBM MQ 9.2, configuration properties for the **mqiptAdmin** command can also be specified in a properties file.

The MQIPT configuration file

The MQIPT configuration file comprises a number of sections. There is one `[global]` section, and an additional `[route]` section for each route through MQIPT that has been defined.

Each section contains name/value property pairs. Some properties can appear only in the `[global]` section, some can appear only in the `[route]` sections, and some can appear both in `[route]` and `[global]` sections. If a property appears in both route and `[global]` sections, the value of the property in the `[route]` section overrides the global value, but only for the route in question. In this way, the `[global]` section can be used to establish the default values to be used for those properties not set in the individual `[route]` sections.

The `[global]` section starts with a line containing the characters `[global]` and ends when the first `[route]` section starts. The `[global]` section must precede all `[route]` sections in the file.

Each `[route]` section starts with a line containing the characters `[route]` and ends when the next `[route]` section starts, or when the end of the configuration file is reached.

Any unrecognized property name is ignored. If a property in a `[route]` section has a recognized name but has an invalid value (for example `MinConnectionThreads=x` or `HTTP=unsure`), that route is disabled (that is, it does not listen for any incoming connections).

Attention: The maximum limit for the number of routes that can be added in the `mqipt.conf` file is 100.

Invalid values for properties in the `[global]` section might prevent MQIPT, or the command server, from starting. If the command server does not start, MQIPT does not listen for administrative commands sent by the **mqiptAdmin** command to the affected command port. If properties with invalid values in the `[global]` section are present when MQIPT is refreshed, a warning message is issued and the effective value of the property remains unchanged. This prevents invalid property values from causing an active instance of MQIPT to shut down when it is refreshed.

Where a property is listed as taking the values `true` or `false`, any mixture of uppercase and lowercase characters can be used in the property value.

You can change the value of a property by editing the `mqipt.conf` file. To apply any changes, refresh MQIPT by using the **mqiptAdmin** command with the **-refresh** keyword.

To include comments in the configuration file, start a line with a "#" character.

Changes to certain properties cause a route to be restarted only if other properties are already enabled. For example, any changes to the HTTP properties have an effect only if the **HTTP** property is also enabled.

When a route is restarted, existing connections are terminated. To override this behavior, set the **RouteRestart** property to `false`. This prevents the route from restarting, allowing existing connections to remain active until the **RouteRestart** property is re-enabled.

For information about how to set up some simple configurations, see Getting started with MQIPT. For a sample configuration, see the `mqiptSample.conf` file in the MQIPT installation directory.

The mqiptAdmin properties file

Configuration properties for the **mqiptAdmin** command can be specified in a separate properties file. These configuration properties are needed when **mqiptAdmin** connects to the MQIPT TLS command port.

For the list of properties that can be specified in the **mqiptAdmin** properties file, see “mqiptAdmin properties” on page 255. Property names are case-sensitive. Any unrecognized properties are ignored.

Comments can be included in the properties file by starting a line with a "#" character.

Summary of MQIPT properties

This table shows a summary of MQIPT configuration properties and includes the following information:

- An alphabetical list of MQIPT properties with links to further information in the `[route]` section, or the `[global]` section if the `[route]` section does not apply.
- The property that must be set to `true` for a value to have an effect.
- Whether the property applies to the `[global]` section, the `[route]` section, or both.
- Default values that are used if a property is missing from both the `[route]` section and the `[global]` section. When specifying the values `true` and `false`, any mixture of uppercase and lowercase characters can be used.

Name of property	Property to set `true`	Global	Route	Default
AccessPW		yes	no	null
Active		yes	yes	`true`
ClientAccess		yes	yes	`false`

Name of property	Property to set `true`	Global	Route	Default
CommandPort		yes	no	null
CommandPortListenerAddress		yes	no	null
ConnectionLog		yes	no	`true`
Destination		no	yes	null
DestinationPort		no	yes	`1414`
“EnableAdvancedCapabilities” on page 234		yes	no	`false`
HTTP		yes	yes	`false`
HTTPProxy	HTTP	yes	yes	null
HTTPProxyPort	HTTP	yes	yes	`8080`
HTTPS	HTTP	yes	yes	`false`
HTTPServer	HTTP	yes	yes	null
HTTPServerPort	HTTP	yes	yes	null
IdleTimeout		yes	yes	`0`
IgnoreExpiredCRLs		yes	yes	`false`
LDAP		yes	yes	`false`
LDAPIgnoreErrors	LDAP	yes	yes	`false`
LDAPCacheTimeout	LDAP	yes	yes	`24`
LDAPServer1	LDAP	yes	yes	null
LDAPServer1Port	LDAP	yes	yes	`389`
LDAPServer1Userid	LDAP	yes	yes	null
LDAPServer1Password	LDAP	yes	yes	null
LDAPServer1Timeout	LDAP	yes	yes	`0`
LDAPServer2	LDAP	yes	yes	null
LDAPServer2Port	LDAP	yes	yes	`389`
LDAPServer2Userid	LDAP	yes	yes	null
LDAPServer2Password	LDAP	yes	yes	null
LDAPServer2Timeout	LDAP	yes	yes	`0`
ListenerAddress		yes	yes	null
ListenerPort		no	yes	null
LocalAddress		yes	yes	null
LocalAdmin		yes	no	`true`
MaxConnectionThreads		yes	yes	`100`
MaxLogFileSize		yes	no	`50`
MinConnectionThreads		yes	yes	`5`

Name of property	Property to set `true`	Global	Route	Default
Name		no	yes	null
OutgoingPort		no	yes	`0`
V 9.3.1 PasswordProtection		yes	yes	`required`
QMgrAccess		yes	yes	`true`
RemoteCommandAuthentication		yes	no	`none`
RemoteShutdown		yes	no	`false`
RouteRestart		yes	yes	`true`
SecurityExit		yes	yes	`false`
SecurityExitName	SecurityExit	yes	yes	null
SecurityExitPath	SecurityExit	yes	yes	*mqipt_home*`\exits`
SecurityExitTimeout	SecurityExit	yes	yes	`30`
SecurityManager (Note 3)		yes	no	`false`
SecurityManagerPolicy (Note 3)		yes	no	null
SocksClient		yes	yes	`false`
SocksProxyHost	SocksClient	yes	yes	null
SocksProxyPort	SocksClient	yes	yes	`1080`
SocksServer		yes	yes	`false`
SSLClient		yes	yes	`false`
SSLClientCAKeyRing	SSLClient	yes	yes	null
SSLClientCAKeyRingPW	SSLClient	yes	yes	null
“SSLClientCAKeyRingUseCryptoHardware” on page 243	SSLClient	yes	yes	`false`
SSLClientCipherSuites	SSLClient	yes	yes	null
SSLClientConnectTimeout	SSLClient	yes	yes	`30`
V 9.3.0 SSLClientCustomOutboundSNI	SSLClient	yes	yes	null
SSLClientDN_C	SSLClient	yes	yes	* (Note 1)
SSLClientDN_CN	SSLClient	yes	yes	* (Note 1)
SSLClientDN_DC	SSLClient	yes	yes	* (Note 1)
SSLClientDN_DNQ	SSLClient	yes	yes	* (Note 1)
SSLClientDN_L	SSLClient	yes	yes	* (Note 1)
SSLClientDN_O	SSLClient	yes	yes	* (Note 1)
SSLClientDN_OU	SSLClient	yes	yes	* (Note 1)
SSLClientDN_PC	SSLClient	yes	yes	* (Note 1)

Name of property	Property to set `true`	Global	Route	Default
SSLClientDN_ST	SSLClient	yes	yes	* (Note 1)
SSLClientDN_Street	SSLClient	yes	yes	* (Note 1)
SSLClientDN_T	SSLClient	yes	yes	* (Note 1)
SSLClientDN_UID	SSLClient	yes	yes	* (Note 1)
SSLClientExit		yes	yes	`false`
SSLClientKeyRing	SSLClient	yes	yes	null
SSLClientKeyRingPW	SSLClient	yes	yes	null
"SSLClientKeyRingUseCryptoHardware" on page 245	SSLClient	yes	yes	`false`
"[MQ 9.3.0 Jun 2022]SSLClientOutboundSNI" on page 246	SSLClient	yes	yes	`hostname`
SSLClientProtocols	SSLClient	yes	yes	V 9.3.0 `TLSv1.2 TLSv1.3`
SSLClientSiteDN_C	SSLClient	yes	yes	* (Note 1)
SSLClientSiteDN_CN	SSLClient	yes	yes	* (Note 1)
SSLClientSiteDN_DC	SSLClient	yes	yes	* (Note 1)
SSLClientSiteDN_DNQ	SSLClient	yes	yes	* (Note 1)
SSLClientSiteDN_L	SSLClient	yes	yes	* (Note 1)
SSLClientSiteDN_O	SSLClient	yes	yes	* (Note 1)
SSLClientSiteDN_OU	SSLClient	yes	yes	* (Note 1)
SSLClientSiteDN_PC	SSLClient	yes	yes	* (Note 1)
SSLClientSiteDN_ST	SSLClient	yes	yes	* (Note 1)
SSLClientSiteDN_Street	SSLClient	yes	yes	* (Note 1)
SSLClientSiteDN_T	SSLClient	yes	yes	* (Note 1)
SSLClientSiteDN_UID	SSLClient	yes	yes	* (Note 1)
SSLClientSiteLabel	SSLClient	yes	yes	null
SSLCommandPort		yes	no	null
SSLCommandPortCipherSuites		yes	no	null
SSLCommandPortListenerAddress		yes	no	null
SSLCommandPortKeyRing		yes	no	null
SSLCommandPortKeyRingPW		yes	no	null
SSLCommandPortKeyRingUseCryptoHardware		yes	no	`false`

Name of property	Property to set `true`	Global	Route	Default

SSLCommandPortProtocols		yes	no	V 9.3.0
				`TLSv1.2`
				`TLSv1.3`

SSLCommandPortSiteLabel		yes	no	null
SSLExitData	SSLServerExit	yes	yes	null
SSLExitName	SSLServerExit	yes	yes	null
SSLExitPath	SSLServerExit	yes	yes	*mqipt_home*`\ exits`
SSLExitTimeout	SSLServerExit	yes	yes	`30`
SSLProxyMode		yes	yes	`false`
SSLPlainConnections	either SSLServer or SSLProxyMode	yes	yes	`false`
SSLServer		yes	yes	`false`
SSLServerAskClientAuth	SSLServer	yes	yes	`false`
SSLServerCAKeyRing	SSLServer	yes	yes	null
SSLServerCAKeyRingPW	SSLServer	yes	yes	null
"SSLServerCAKeyRingUseCryptoHardware" on page 249	SSLServer	yes	yes	`false`
SSLServerCipherSuites	SSLServer	yes	yes	null
SSLServerDN_C	SSLServer	yes	yes	* (Note 1)
SSLServerDN_CN	SSLServer	yes	yes	* (Note 1)
SSLServerDN_DC	SSLServer	yes	yes	* (Note 1)
SSLServerDN_DNQ	SSLServer	yes	yes	* (Note 1)
SSLServerDN_L	SSLServer	yes	yes	* (Note 1)
SSLServerDN_O	SSLServer	yes	yes	* (Note 1)
SSLServerDN_OU	SSLServer	yes	yes	* (Note 1)
SSLServerDN_PC	SSLServer	yes	yes	* (Note 1)
SSLServerDN_ST	SSLServer	yes	yes	* (Note 1)
SSLServerDN_Street	SSLServer	yes	yes	* (Note 1)
SSLServerDN_T	SSLServer	yes	yes	* (Note 1)
SSLServerDN_UID	SSLServer	yes	yes	* (Note 1)
SSLServerExit		yes	yes	`false`
SSLServerKeyRing	SSLServer	yes	yes	null
SSLServerKeyRingPW	SSLServer	yes	yes	null
"SSLServerKeyRingUseCryptoHardware" on page 252	SSLServer	yes	yes	`false`

Name of property	Property to set `true`	Global	Route	Default
SSLServerProtocols	SSLServer	yes	yes	V 9.3.0 `TLSv1.2 TLSv1.3`
SSLServerSiteDN_C	SSLServer	yes	yes	* (Note 1)
SSLServerSiteDN_CN	SSLServer	yes	yes	* (Note 1)
SSLServerSiteDN_DC	SSLServer	yes	yes	* (Note 1)
SSLServerSiteDN_DNQ	SSLServer	yes	yes	* (Note 1)
SSLServerSiteDN_L	SSLServer	yes	yes	* (Note 1)
SSLServerSiteDN_O	SSLServer	yes	yes	* (Note 1)
SSLServerSiteDN_OU	SSLServer	yes	yes	* (Note 1)
SSLServerSiteDN_PC	SSLServer	yes	yes	* (Note 1)
SSLServerSiteDN_ST	SSLServer	yes	yes	* (Note 1)
SSLServerSiteDN_Street	SSLServer	yes	yes	* (Note 1)
SSLServerSiteDN_T	SSLServer	yes	yes	* (Note 1)
SSLServerSiteDN_UID	SSLServer	yes	yes	* (Note 1)
SSLServerSiteLabel	SSLServer	yes	yes	null
StoredCredentialsFormat		yes	yes	null
TCPKeepAlive		yes	yes	false
Trace		yes	yes	`0`
V 9.3.2 TraceFileCount		yes	no	25
V 9.3.2 TraceFileSize		yes	no	`200`
"[MQ 9.3.0 Jun 2022][MQ 9.3.0 Jun 2022]TraceUserData" on page 254		yes	yes	64
UriName	HTTP	yes	yes	(Note 2)

Notes:

1. The asterisk (*) represents a wildcard.
2. See UriName in "MQIPT route properties" on page 237 for details about the default settings.
3. Deprecated This property is deprecated for removal in a future release.

Related reference

"IBM MQ Internet Pass-Thru configuration reference" on page 227
IBM MQ Internet Pass-Thru (MQIPT) uses a configuration file called `mqipt.conf` to define routes and to control the actions of the MQIPT server. From IBM MQ 9.2, configuration properties for the **`mqiptAdmin`** command can also be specified in a properties file.

"MQIPT global properties" on page 234
The `mqipt.conf` configuration file can contain a number of global properties.

"MQIPT route properties" on page 237

The `mqipt.conf` configuration file can contain properties for individual routes.

MQIPT global properties

The `mqipt.conf` configuration file can contain a number of global properties.

The following properties can appear only in the `[global]` section of `mqipt.conf`. All the route properties except **ListenerPort**, **Destination**, **DestinationPort**, **Name**, and **OutgoingPort** can also appear in the `[global]` section. If a property appears in both route and `[global]` sections, the value of the property in the `[route]` section overrides the global value, but only for the route in question. In this way, the `[global]` section can be used to establish the default values to be used for those properties not set in the individual `[route]` sections.

AccessPW
The password used to authenticate commands sent to the MQIPT command port using the **mqiptAdmin** command.

The value can be either a password that has been encrypted using the **mqiptPW** command, or a plain text password. Plain text passwords can only contain alphanumeric characters. You are strongly encouraged to encrypt passwords that are stored in the MQIPT configuration. For more information on encrypting passwords in the MQIPT configuration, see Encrypting stored passwords.

Authentication is performed for administrative commands received by the command port if both of the following conditions are true:

- The **AccessPW** property is specified and set to a value that is not blank.
- The **RemoteCommandAuthentication** property is specified and set to a value other than `none`.

CommandPort
The TCP/IP port number of the unsecured command port. MQIPT accepts administrative commands that are sent by the **mqiptAdmin** command to this command port.

Connections to the unsecured command port are not secured with TLS. Data sent to the command port, including the access password, might be accessed by other users of the network. To configure a command port that is secured with TLS, set the **SSLCommandPort** property instead.

If the **CommandPort** property is not specified, MQIPT does not listen for administrative commands on the unsecured command port. To use the default port number, 1881, used by default by the **mqiptAdmin** command, set **CommandPort** to 1881.

CommandPortListenerAddress
The local listener address to be used by the unsecured command port. By setting the local listener address you can restrict inbound connections to the unsecured command port to those from a particular network interface. The default is to listen on all network interfaces.

ConnectionLog
Either `true` or `false`. When true, MQIPT logs all connection attempts (successful or otherwise) in the `logs` subdirectory and disconnection events to the file `mqipt`*YYYYMMDDHHmmSS*`.log` (where *YYYYMMDDHHmmSS* are characters representing the current date and time). The default value of **ConnectionLog** is `true`. When this property is changed from `true` to `false`, MQIPT closes the existing connection log and creates a new one. The new log is used when the property is reset to `true`.

EnableAdvancedCapabilities
Set this property to `true` to confirm that advanced capabilities that require IBM MQ Advanced, IBM MQ Appliance, IBM MQ Advanced for z/OS, or IBM MQ Advanced for z/OS VUE entitlement can be used by MQIPT. If you have appropriate entitlement you can use the advanced capabilities in MQIPT. If advanced capabilities are enabled on a route, the local queue manager that is connected using the MQIPT route is also required to have IBM MQ Advanced, IBM MQ Appliance, IBM MQ Advanced for z/OS, or IBM MQ Advanced for z/OS VUE entitlement. Routes that use advanced capabilities cannot start unless this property is set to `true`. When this property is changed from `true` to `false`, routes that use advanced capabilities are stopped.

LocalAdmin
Specifies whether local administration without a command port is permitted. Administrative commands sent by the **mqiptAdmin** command using local administration instead of the command port, are not accepted if this property is set to `false`.

Valid values for this property are `true` and `false`. The default value is `true`.

MaxLogFileSize
The maximum size (specified in KB) of the connection log file. When the file size increases above this maximum a backup copy (`mqipt001.log`) is made, and a new file is started. Only two backup files are kept (`mqipt001.log` and `mqipt002.log`); each time the main log file fills up, any earlier backups are erased. The default value of **MaxLogFileSize** is 50; the minimum allowed value is 5.

RemoteCommandAuthentication
Specifies whether administrative commands received by the unsecured command port or TLS command port should be authenticated. Commands are authenticated by checking that the password supplied matches the password specified in the `AccessPW` property. The value can be one of the following values:

none
No authentication is performed on commands issued to either of the command ports. Users of the **mqiptAdmin** command do not need to enter a password. This is the default value.

optional
Users of the **mqiptAdmin** command are not required to provide a password. However, if a password is provided it must be valid.

required
Users of the **mqiptAdmin** command are required to provide a valid password with every command issued to the command ports.

The **AccessPW** property must also be specified to enable authentication for the command ports.

RemoteShutDown
Specifies whether MQIPT can be shut down by a stop command sent to the unsecured command port or the TLS command port by the **mqiptAdmin** command. This property must be set to `true` for stop commands received by either of the command ports to be processed.

Valid values for this property are `true` and `false`. The default value is `false`.

SecurityManager
Set this property to `true` to enable the Java security manager for this instance of MQIPT. You must ensure that the correct permissions are granted. See Java security manager for more information. The default value for this property is `false`.

This property is deprecated for removal in a future release.

SecurityManagerPolicy
The fully-qualified file name of a Java security manager policy file. If this property is not set then only the default system and user policy files are used. If the Java security manager is already enabled, then changes to this property have no effect until the Java security manager has been disabled and re-enabled.

This property is deprecated for removal in a future release.

SSLCommandPort
The TCP/IP port number of the TLS command port. MQIPT accepts administrative commands that are sent by the **mqiptAdmin** command to this command port. This port only accepts TLS connections. This property must be specified in order to enable the TLS command port.

SSLCommandPortCipherSuites
The name of the cipher suites to enable on the TLS command port. More than one cipher suite can be specified by separating the values with commas. Only TLS 1.2 and TLS 1.3 cipher suites that are enabled by default in the Java runtime environment (JRE) supplied with MQIPT can be specified. If this property is not specified, all cipher suites that are enabled in the JRE are enabled on the TLS command port.

SSLCommandPortListenerAddress
The local listener address to be used by the TLS command port. By setting the local listener address you can restrict inbound connections to the TLS command port to those from a particular network interface. The default is to listen on all network interfaces.

SSLCommandPortKeyRing
The name of the PKCS#12 key ring file that contains the TLS command port server certificate.

On Windows platforms, you must use a double backslash (\\) as the file separator.

SSLCommandPortKeyRingPW
The encrypted password to access the TLS command port key ring file or the PKCS #11 key store. The password must be encrypted using the `mqiptPW` command, and the value of this property set to the string output by `mqiptPW`.

SSLCommandPortKeyRingUseCryptoHardware
Specifies whether cryptographic hardware that supports the PKCS #11 interface is used as the key store for the TLS command port server certificate. Valid values for this property are `true` and `false`. If this property is set to `true`, the **SSLCommandPortKeyRing** cannot also be specified.

Use of cryptographic hardware in MQIPT is an IBM MQ Advanced feature. The `EnableAdvancedCapabilities` property must be set to `true` to confirm that you have IBM MQ Advanced entitlement.

SSLCommandPortProtocols
A comma-separated list of protocols to enable on the TLS command port. One or more of the following values can be specified.

Table 117. Permitted values for command port TLS protocols

Value	Protocol
TLSv1.2	TLS 1.2
V 9.3.0 TLSv1.3	TLS 1.3

In versions earlier than IBM MQ 9.2.5, if you do not specify this property, the only protocol enabled by default is TLS 1.2. From IBM MQ 9.2.5, if you do not specify this property, TLS 1.2 and TLS 1.3 are enabled by default.

SSLCommandPortSiteLabel
The label name of the server certificate used by the TLS command port. If this property is not specified, any certificate in the TLS command port key store that is compatible with the cipher suite is selected.

Trace

The level of trace for global MQIPT threads that are not associated with a route, and for routes that have no **Trace** property set. For example, the main MQIPT control thread and the command server threads are not associated with a route and are only traced if trace is enabled in the `[global]` section. The value of the **Trace** property in a `[route]` section overrides the global **Trace** property, for that route. For information about tracing threads associated with a route, see **Trace** in the `[route]` section.

The value of this property can be one of the following:

0
Trace is not enabled

Any positive integer
Trace is enabled

The default value is `0`.

V 9.3.2 **TraceFileCount**

The number of trace files in the rotating set of files used by MQIPT to write trace data.

The minimum allowed value is 3. The default value is 25.

If you change the value of this property, the current trace file is closed, and the next file in the rotating set of trace files is opened.

V 9.3.2 **TraceFileSize**

The maximum size of the trace files produced by MQIPT, specified in MB.

The minimum allowed value is 1. The default value is 200.

If you change the value of this property, the current trace file is closed, and the next file in the rotating set of trace files is opened.

MQIPT route properties

The `mqipt.conf` configuration file can contain properties for individual routes.

The `[route]` section of the `mqipt.conf` configuration file can contain the following properties:

Active
The route accepts incoming connections only if the value of **`Active`** is set to `true`. This means that you can temporarily shut off access to the destination, by setting this value to `false`, without having to delete the `[route]` section from the configuration file. If you change this property to `false`, the route is stopped when a refresh command is issued. All connections to the route are stopped.

ClientAccess
The route allows incoming client channel connections only if the value of **`ClientAccess`** is set to `true`. Note that potentially you can configure MQIPT to accept client requests only, queue manager requests only, or both types of request. Use this property in conjunction with the **`QMgrAccess`** property. If you change this property to `false`, the route is stopped, and restarted when a refresh command is issued. All connections to the route are stopped.

Destination
The host name (or dotted decimal IP address) of the queue manager, or subsequent MQIPT instance, to which this route is to connect. Each `[route]` section must contain an explicit **`Destination`** value, but several `[route]` sections can refer to the same destination. If a change to this property affects a route, the route is stopped, and restarted when a refresh command is issued. All connections to the route are stopped. When using the **`SocksProxyHost`** property the **`Destination`** property must use the dotted decimal IPv4 address format.

DestinationPort
The port on the destination host to which this route is to connect. Each `[route]` section must contain an explicit **`DestinationPort`** value, but several routes can refer to the same combination of **`Destination`** and **`DestinationPort`**values. If a change to this property affects a route, the route is stopped, and restarted when a refresh command is issued. All connections to the route are stopped.

HTTP
Set **`HTTP`** to `true` for routes responsible for making outbound HTTP tunneling requests. The **`Destination`** property for the route must be the host name of another MQIPT when HTTP it set to true. Set **`HTTP`** to `false` for routes connected to IBM MQ queue managers. If you change this property, the route is stopped. At least one of the **`HTTPProxy`** or **`HTTPServer`** properties must also be specified when HTTP is set to true. This property cannot be used in conjunction with the **`SocksClient`** property.

HTTPProxy
The host name (or dotted decimal IP address) of the HTTP proxy used by all connections for this route. A **`CONNECT`** request is issued to the HTTP proxy, instead of the **`POST`** request that is normally used when no HTTP proxy is configured. If you change this property (and **`HTTP`** is set to `true`), the route is stopped, and restarted when a refresh command is issued. All connections to the route are stopped.

HTTPProxyPort
The port address to use on the HTTP proxy. The default value is 8080. If you change this property (and **HTTP** is set to true), the route is stopped, and restarted when a refresh command is issued. All connections to the route are stopped.

HTTPServer
The host name (or dotted decimal IP address) of the HTTP server used by all connections for this route. This is usually the host name of another MQIPT.

If **HTTPProxy** is not specified, MQIPT connects to the host specified in **HTTPServer**, and issues HTTP **POST** requests to the host specified in the route **Destination** property. If **HTTPProxy** is specified, MQIPT connects to the host specified in **HTTPProxy** instead, and requests that the proxy establish a tunnel to the host specified in **HTTPServer**.

If **HTTPProxy** is specified, the default value is the route **Destination**.

If you change this property (and **HTTP** is set to true), the route is stopped, and restarted when a refresh command is issued. All connections to the route are stopped.

HTTPS
Set **HTTPS** to true to make HTTPS requests. The **HTTP** and **SSLClient** properties must also be enabled, and the client key ring configured using the **SSLClientKeyRing** or **SSLClientKeyRingUseCryptoHardware** property, as for SSL/TLS operation. If you change the **HTTPS** property (and **HTTP** is set to true), the route is stopped, and restarted when a refresh command is issued. All connections to the route are stopped.

HTTPServerPort
The port address to use on the HTTP server. The default value is 8080, unless **HTTPProxy** is specified, in which case the default value is the route **DestinationPort**.

If you change this property (and **HTTP** is set to true), the route is stopped, and restarted when a refresh command is issued. All connections to the route are stopped.

IdleTimeout
The time, in minutes, after which an idle connection is closed. Note that queue manager to queue manager channels also have the **DISCINT** property. If you set the **IdleTimeout** parameter, take note of **DISCINT**. If **IdleTimeout** is set to 0, there is no idle timeout. Changes to this property take effect only when the route is restarted.

IgnoreExpiredCRLs
Set **IgnoreExpiredCRLs** to true to ignore an expired CRL. The default value is false. Note that if you set **IgnoreExpiredCRLs** to true, a revoked certificate could be used to make an SSL/TLS connection.

LDAP
Set **LDAP** to true to enable use of an LDAP server when using SSL/TLS connections. MQIPT will use the LDAP server to retrieve CRLs and ARLs. The **SSLClient** property or **SSLServer** property must also be set to true for this property to take effect.

LDAPCacheTimeout
The expiry time, in hours, of the temporary cache in which a CRL retrieved from an LDAP server, is stored. After this time, the entire CRL cache is emptied. For example, specifying a value of 1 hour means that the cache is emptied once per hour. The default value is 24. If you specify a timeout value of 0, entries in the cache will not expire until the route is restarted. If you change this property (and **LDAP** is set to true), the route is stopped, and restarted when a refresh command is issued. All connections to the route are stopped.

LDAPIgnoreErrors
Set **LDAPIgnoreErrors** to true to ignore any connection or timeout errors when performing an LDAP search. If MQIPT cannot perform a successful search, it will not allow the client connection to complete, unless this property has been enabled. A successful search means that a CRL has been retrieved or there are no CRLs available for the specified CA. If you change this property (and **LDAP** is set to true), the route is stopped, and restarted when a refresh command is issued. All connections to the route are stopped.

Note: If you enable this property, a revoked certificate could be used to make an SSL/TLS connection.

LDAPServer1
The host name or IP address of the main LDAP server. This property must be set if LDAP has been set to `true`. If you change this property (and **LDAP** is set to `true`), the route is stopped, and restarted when a refresh command is issued. All connections to the route are stopped.

LDAPServer1Port
The listening port number of the main LDAP server. The default value is 389. If you change this property (and **LDAP** is set to `true`), the route is stopped, and restarted when a refresh command is issued. All connections to the route are stopped.

LDAPServer1Userid
The user ID needed to access the main LDAP server. This property must be set if authorization to access the main LDAP server is required. If you change this property (and **LDAP** is set to `true`), the route is stopped, and restarted when a refresh command is issued. All connections to the route are stopped.

LDAPServer1Password
The password needed to access the main LDAP server. This property must be set if **LDAPServer1Userid** has been set to `true`. If you change this property (and **LDAP** is set to `true`), the route is stopped, and restarted when a refresh command is issued. All connections to the route are stopped.

The value can be either a password that has been encrypted using the **mqiptPW** command, or a plain text password. Plain text passwords can only contain alphanumeric characters. You are strongly encouraged to encrypt passwords that are stored in the MQIPT configuration. For more information on encrypting passwords in the MQIPT configuration, see Encrypting stored passwords.

LDAPServer1Timeout
The time, in seconds, that MQIPT waits for a response from the main LDAP server. The default value is `0`, which means the connection will not time out. If you change this property (and **LDAP** is set to `true`), the route is stopped, and restarted when a refresh command is issued. All connections to the route are stopped.

LDAPServer2
The host name or IP address of the backup LDAP server. This property is optional. If you change this property (and **LDAP** is set to `true`), the route is stopped, and restarted when a refresh command is issued. All connections to the route are stopped.

LDAPServer2Port
The listening port number of the backup LDAP server. The default value is 389. If you change this property (and **LDAP** is set to `true`), the route is stopped, and restarted when a refresh command is issued. All connections to the route are stopped.

LDAPServer2Userid
The userid needed to access the backup LDAP server. This property must be set if authorization to access the backup LDAP server is required. If you change this property (and **LDAP** is set to `true`), the route is stopped, and restarted when a refresh command is issued. All connections to the route are stopped.

LDAPServer2Password
The password needed to access the backup LDAP server. This property must be set if **LDAPServer2** has been set to `true`. If you change this property (and **LDAP** is set to `true`), the route is stopped, and restarted when a refresh command is issued. All connections to the route are stopped.

The value can be either a password that has been encrypted using the **mqiptPW** command, or a plain text password. Plain text passwords can only contain alphanumeric characters. You are strongly encouraged to encrypt passwords that are stored in the MQIPT configuration. For more information on encrypting passwords in the MQIPT configuration, see Encrypting stored passwords.

LDAPServer2Timeout
The time, in seconds, that MQIPT will wait for a response from the backup LDAP server. The default value is `0`, which means the connection will not time out. If you change this property (and **LDAP** is set to `true`), the route is stopped, and restarted when a refresh command is issued. All connections to the route are stopped.

ListenerAddress
Use this property if the MQIPT system has multiple IP addresses and you need to bind the route listener port to a specific address. This is useful for restricting inbound connections to those from a particular network interface. The value of this property should be an IP address belonging to one of the network interfaces on the system where MQIPT is running. The default is to accept connections from all network interfaces.

ListenerPort
The port number on which the route should listen for incoming requests. Each `[route]` section must contain an explicit **`ListenerPort`** value The **`ListenerPort`** values set in each section must be distinct. Any valid port number can be used, including ports `80` and 443, provided that the ports chosen are not already in use by any other TCP/IP listener running on the same host.

LocalAddress
The IP address to bind all connections to for this route on this computer. The chosen address must be an IP address that is associated with one of the network interfaces on the computer on which MQIPT is running. If you change this property, the route is stopped, and restarted when a refresh command is issued. All connections to the route are stopped.

MaxConnectionThreads
The maximum number of connection threads, and thus the maximum number of concurrent connections, that can be handled by this route. If this limit is reached, the **`MaxConnectionThreads`** value also indicates the number of connections that are queued when all the threads are in use. Beyond that number, subsequent connection requests are refused.

The minimum allowed value is the greater of 1 and the value of **`MinConnectionThreads`**.

If the value is increased, the new value is used when the refresh command is issued. All connections use the new value immediately. The route is not stopped.

If the value is decreased, the new value takes effect only when the route is restarted.

MinConnectionThreads
The number of connection threads allocated to handle incoming connections on a route when the route is started. The number of threads allocated does not drop below this value during the time the route is active.

The value must be in the range `0` to the value of **`MaxConnectionThreads`**.

Changes to this property take effect only when the route is restarted.

Name
A name to help identify the route. This property is optional. The value is shown in console messages and tracing information. Changes to this property take effect only when the route is restarted.

OutgoingPort
The starting port number used by outgoing connections. The range of port numbers match the **`MaxConnectionThread`** value for this route. The default value of `0` uses a system-defined port number. If you change this property, the route is stopped and restarted when a refresh command is issued. All connections to this route are stopped. When HTTP is used, each channel connection requires two outgoing ports. For more information, see Port number control.

V 9.3.1 **PasswordProtection**
Specifies whether MQIPT can add or remove protection for credentials sent in MQCSP structures by IBM MQ clients, in order to maintain compatibility between the client and queue manager, for MQIPT routes that are configured to add or remove TLS encryption.

Credentials in MQCSP structures can either be protected, by using the IBM MQ MQCSP password protection feature, or encrypted by using TLS encryption. MQCSP password protection is useful for test and development purposes as it is simpler than setting up TLS encryption, but it is not as secure.

For more information about MQCSP password protection, see MQCSP password protection.

When an MQIPT route is configured to add or remove TLS encryption, MQIPT might need to protect the credentials in the MQCSP structure, or remove the MQCSP password protection, for the connection to be successful.

The value of the property can be one of the following values:

required
MQIPT ensures that credentials in the MQCSP structure are either encrypted by using TLS or protected with MQCSP password protection.

If credentials in the MQCSP structure are sent encrypted by the client by using TLS encryption, and the MQIPT route removes the TLS encryption, MQIPT protects the credentials with MQCSP password protection before forwarding the credentials to the route destination. This occurs when the MQIPT route is configured with `SSLServer=true` and `SSLClient=false`, and the selected CipherSuite does not use a null cipher.

If credentials in the MQCSP structure are protected by the client with MQCSP password protection, MQIPT does not remove the protection, even if the connection between MQIPT and the route destination uses TLS encryption. If the connection between MQIPT and the route destination uses TLS encryption, the connection might fail with reason code MQRC_PASSWORD_PROTECTION_ERROR (2594).

This is the default value.

compatible
MQIPT applies or removes MQCSP password protection as required to ensure that the connection is successful.

If credentials in the MQCSP structure are sent encrypted by the client by using TLS encryption, and the MQIPT route removes the TLS encryption, MQIPT protects the credentials with MQCSP password protection before forwarding the password to the route destination. This occurs when the MQIPT route is configured with `SSLServer=true` and `SSLClient=false`, and the selected CipherSuite does not use a null cipher.

If credentials in the MQCSP structure are protected by the client with MQCSP password protection, and the MQIPT route add TLS encryption, MQIPT removes the MQCSP password protection before forwarding the credentials to the route destination. This occurs when the MQIPT route is configured with `SSLServer=false` and `SSLClient=true`, and the selected CipherSuite does not use a null cipher.

This option offers the best compatibility. However, it should only be used for test and development purposes on trusted networks, as it does not ensure that the password is protected on the network.

passthru
Credentials in the MQCSP structure are forwarded to the route destination by MQIPT without adding or removing MQCSP password protection. If the MQIPT route is configured to add or remove TLS encryption, client connections might fail with reason code MQRC_PASSWORD_PROTECTION_ERROR (2594).

QMgrAccess
Set **`QMgrAccess`** to `true` to allow incoming queue manager channel connections (for example sender channels). If you change this property to `false`, the route is stopped when a refresh command is issued. All connections to this route are stopped.

RouteRestart
Set **`RouteRestart`** to `false` to stop the route from restarting when other route properties have been changed and a refresh command has been issued. The default value for this property is `true`.

SecurityExit
Set **`SecurityExit`** to `true` to enable a user-defined security exit. The default value for this property is `false`.

SecurityExitName
The class name of the user-defined security exit. This property must be set if **`SecurityExit`** has been set to `true`. If you change this property (and **`SecurityExit`** is set to `true`), the route is stopped, and restarted when a refresh command is issued. All connections to this route are stopped.

SecurityExitPath
The fully-qualified path name containing the user-defined security exit. If this property has not been set, then it will default to the exits subdirectory. This property can also define the name of

a Java archive (JAR) file containing the user-defined security exit. If you change this property (and **SecurityExit** is set to `true`), the route is stopped and restarted when a refresh command is issued. All connections to this route are stopped.

SecurityExitTimeout
The timeout value (in seconds) used by MQIPT to determine how long to wait for a response when validating a connection request. The default value is `30`. If you change this property (and **SecurityExit** is set to `true`), the route is stopped, and restarted when a refresh command is issued. All connections to the route are stopped.

SocksClient
Set **SocksClient** to `true` to make the route act as a SOCKS client and define all connections through the SOCKS proxy with the **SocksProxyHost** and **SocksProxyPort** properties. If you change this property, the route is stopped, and restarted when a refresh command is issued. All connections to the route are stopped. This property cannot be used with:

- **HTTP**
- **SocksServer**
- **SSLClient**
- **SSLProxyMode**

SocksProxyHost
The host name (or dotted decimal IPv4 address) of the SOCKS proxy that all connections for this route use. If you change this property (and **SocksClient** is set to `true`), the route is stopped, and restarted when a refresh command is issued. All connections to this route are stopped. When using the **SocksProxyHost** property the **Destination** property must use the dotted decimal format.

SocksProxyPort
The port number to use on a SOCKS proxy. The default value is `1080`. If you change this property (and **SocksClient** is set to `true`), the route is stopped, and restarted when a refresh command is issued. All connections to the route are stopped.

SocksServer
Set **SocksServer** to `true` to make the route act as a SOCKS proxy and accept SOCKS client connections. If you change this property, the route is stopped, and restarted when a refresh command is issued. All connections to the route are stopped. This property cannot be used with the following properties:

- **SocksClient**
- **SSLProxyMode**
- **SSLServer**

SSLClient
Set **SSLClient** to `true` to make the route act as an SSL/TLS client and make outgoing SSL/TLS connections. Setting **SSLClient** to `true` implies that the destination is either another instance of MQIPT acting as an SSL/TLS server, or an HTTP proxy/server.

If you set **SSLClient** to `true`, you must specify a SSL/TLS client key ring using the **SSLClientKeyRing** or **SSLClientCAKeyRing** property, or configure MQIPT to use cryptographic hardware by setting the **SSLClientKeyRingUseCryptoHardware** or **SSLClientCAKeyRingUseCryptoHardware** property.

If you change **SSLClient**, the route is stopped, and restarted when a refresh command is issued. All connections to this route are stopped.

This property cannot be used in conjunction with the following property:

- **SSLProxyMode**

SSLClientCAKeyRing
The fully-qualified file name of the key ring file containing CA certificates, used to authenticate certificates from the SSL/TLS server. On Windows platforms, you must use a double backslash (\\) as

the file separator. If you change this property (and **SSLClient** is set to `true`), the route is stopped, and restarted when a refresh command is issued. All connections to the route are stopped.

SSLClientCAKeyRingPW
The password to open the SSL/TLS client CA key ring file specified with the **SSLClientCAKeyRing** property, or to connect to the cryptographic hardware key store if the **SSLClientCAKeyRingUseCryptoHardware** property is set to `true`.

The value can be either a password that has been encrypted using the **mqiptPW** command, or the fully-qualified file name of the file containing an encrypted password. If you specify a file name on Windows platforms, you must use a double backslash (\\) as the file separator. You are encouraged to migrate any key ring passwords currently stored in a file to use the latest and most secure protection method, by re-encrypting the passwords using the **mqiptPW** utility. For more information on encrypting passwords in the MQIPT configuration, see Encrypting stored passwords.

If you change this property (and **SSLClient** is set to `true`), the route is stopped, and restarted when a refresh command is issued. All connections to the route are stopped.

SSLClientCAKeyRingUseCryptoHardware

Specifies whether cryptographic hardware that supports the PKCS #11 interface is used as the key store for CA certificates used to authenticate server certificates from the SSL/TLS server, when MQIPT is acting as a SSL/TLS client. If this property is set to `true`, **SSLClientCAKeyRing** cannot be set on the same route.

If you change this property (and **SSLClient** is set to `true`), the route is stopped, and restarted when a refresh command is issued. All connections to the route are stopped.

Use of cryptographic hardware with MQIPT is an IBM MQ Advanced capability. To use this capability, the local queue manager that is connected using the MQIPT route is also required to have IBM MQ Advanced, IBM MQ Appliance, IBM MQ Advanced for z/OS, or IBM MQ Advanced for z/OS VUE entitlement. The route will not start when this property is set to `true` unless the **EnableAdvancedCapabilities** global property is set to confirm that IBM MQ Advanced capabilities can be used.

SSLClientCipherSuites
The name of the SSL/TLS CipherSuite to use on the SSL/TLS client side. This can be one or more of the supported CipherSuites. If you leave this property blank, any CipherSuite for the enabled protocols that is compatible with the client certificate in the key ring is used. If you change this property (and **SSLClient** is set to `true`), the route is stopped, and restarted when a refresh command is issued. All connections to this route are stopped.

SSLClientConnectTimeout
The time (in seconds) that an SSL/TLS client waits for an SSL/TLS connection to be accepted. If you change this property (and **SSLClient** is set to `true`), the route is stopped, and restarted when a refresh command is issued. All connections to the route are stopped.

V 9.3.0 **SSLClientCustomOutboundSNI**
Specifies the value of the Server Name Indication (SNI) when MQIPT initiates a TLS connection to the route destination, if the route is configured with **SSLClientOutboundSNI** set to `custom`. Use this property to set the SNI to a specific value that cannot be set automatically by MQIPT. For example, if you want to set the SNI to a hostname, but the route destination is configured with an IP address.

The value must be a valid Internationalized Domain Name (IDN) compliant with the RFC 3490 specification and cannot end with a trailing dot. The route does not start if an invalid value is specified.

If you change the value of this property, and **SSLClientOutboundSNI** is set to `custom`, the route is stopped and restarted when a refresh command is issued.

Attention: You must not use this setting when forwarding connections to an IBM MQ channel that has a certificate label configured in the channel **CERTLABL** field. If you forward a client in such a way, it will be rejected with a MQRC_SSL_INITIALIZATION_ERROR return code, and an AMQ9673 error printed in the remote queue manager error logs.

SSLClientDN_C
Use this property to accept certificates received from the SSL/TLS server that match this country name. The name can be prefixed or suffixed with an asterisk (*) to extend its scope. Certificate matching is not case sensitive. If you do not specify this property, all country names are accepted. If you change this property (and **SSLClient** is set to `true`), the route is stopped, and restarted when a refresh command is issued. All connections to the route are stopped.

SSLClientDN_CN
Use this property to accept certificates received from the SSL/TLS server that match this common name. The name can be prefixed or suffixed with an asterisk (*) to extend its scope. Certificate matching is not case sensitive. If you do not specify this property, all common names are accepted. If you change this property (and **SSLClient** is set to `true`), the route is stopped, and restarted when a refresh command is issued. All connections to the route are stopped.

SSLClientDN_DC
Use this property to accept certificates received from the SSL/TLS server that match this domain component. The name can be prefixed or suffixed with an asterisk (*) to extend its scope. Certificate matching is not case sensitive. You can specify multiple DCs by separating them with commas. Each DC represents an element in a domain name, for example the domain name `example.ibm.com` is represented as `example,ibm,com` using commas to separate the multiple values. If you do not specify this property, all domain components are accepted. If you change this property (and **SSLClient** is set to `true`), the route is stopped, and restarted when a refresh command is issued. All connections to the route are stopped.

SSLClientDN_DNQ
Use this property to accept certificates received from the SSL/TLS server that match this domain qualifier. The name can be prefixed or suffixed with an asterisk (*) to extend its scope. Certificate matching is not case sensitive. If you do not specify this property, all domain qualifiers are accepted. If you change this property (and **SSLClient** is set to `true`), the route is stopped, and restarted when a refresh command is issued. All connections to the route are stopped.

SSLClientDN_L
Use this property to accept certificates received from the SSL/TLS server that match this location. The name can be prefixed or suffixed with an asterisk (*) to extend its scope. Certificate matching is not case sensitive. If you do not specify this property, you imply "all locations". If you change this property (and **SSLClient** is set to `true`), the route is stopped, and restarted when a refresh command is issued. All connections to the route are stopped.

SSLClientDN_O
Use this property to accept certificates received from the SSL/TLS server that match this organization. The name can be prefixed or suffixed with an asterisk (*) to extend its scope. Certificate matching is not case sensitive. If you do not specify this property, certificates are accepted from all organizations. If you change this property (and **SSLClient** is set to `true`), the route is stopped, and restarted when a refresh command is issued. All connections to the route are stopped.

SSLClientDN_OU
Use this property to accept certificates received from the SSL/TLS server that match this Organizational Unit (OU). The name can be prefixed or suffixed with an asterisk (*) to extend its scope. You can specify multiple OUs by separating them with commas. (Match a literal comma by prefixing it with a backslash (\) character.) Certificate matching is not case sensitive. If you do not specify this property, certificates are accepted with any OU name. If you change this property (and **SSLClient** is set to `true`), the route is stopped and restarted when a refresh command is issued. All connections to this route are stopped.

SSLClientDN_PC
Use this property to accept certificates received from the SSL/TLS server that match this postal code. The name can be prefixed or suffixed with an asterisk (*) to extend its scope. Certificate matching is not case sensitive. If you do not specify this property, all postal codes are accepted. If you change this property (and **SSLClient** is set to `true`), the route is stopped, and restarted when a refresh command is issued. All connections to the route are stopped.

SSLClientDN_ST
Use this property to accept certificates received from the SSL/TLS server that match this state. The name can be prefixed or suffixed with an asterisk (*) to extend its scope. Certificate matching is not case sensitive. If you do not specify this property, certificates are accepted from servers in all states. If you change this property (and **SSLClient** is set to `true`), the route is stopped, and restarted when a refresh command is issued. All connections to the route are stopped.

SSLClientDN_Street
Use this property to accept certificates received from the SSL/TLS server that match this street name. The name can be prefixed or suffixed with an asterisk (*) to extend its scope. Certificate matching is not case sensitive. If you do not specify this property, all street names are accepted. If you change this property (and **SSLClient** is set to `true`), the route is stopped, and restarted when a refresh command is issued. All connections to the route are stopped.

SSLClientDN_T
Use this property to accept certificates received from the SSL/TLS server that match this title. The name can be prefixed or suffixed with an asterisk (*) to extend its scope. Certificate matching is not case sensitive. If you do not specify this property, all titles are accepted. If you change this property (and **SSLClient** is set to `true`), the route is stopped, and restarted when a refresh command is issued. All connections to the route are stopped.

SSLClientDN_UID
Use this property to accept certificates received from the SSL/TLS server that match this user ID. The name can be prefixed or suffixed with an asterisk (*) to extend its scope. Certificate matching is not case sensitive. If you do not specify this property, all user IDs are accepted. If you change this property (and **SSLClient** is set to `true`), the route is stopped, and restarted when a refresh command is issued. All connections to the route are stopped.

SSLClientExit
Use this property to enable or disable the use of an exit when the route is acting as an SSL/TLS client. This allows you to define exit details in the configuration file without them actually being used.

SSLClientKeyRing
The fully-qualified file name of the key ring file containing the client certificate. On Windows platforms, you must use a double backslash (\\) as the file separator. If you change **SSLClientKeyRing** (and **SSLClient** is set to `true`), the route is stopped, and restarted when a refresh command is issued. All connections to the route are stopped.

SSLClientKeyRingPW
The password to open the SSL/TLS client key ring file specified with the **SSLClientKeyRing** property, or to connect to the cryptographic hardware key store if the **SSLClientKeyRingUseCryptoHardware** property is set to `true`.

The value can be either a password that has been encrypted using the **mqiptPW** command, or the fully-qualified file name of the file containing an encrypted password. If you specify a file name on Windows platforms, you must use a double backslash (\\) as the file separator. You are encouraged to migrate any key ring passwords currently stored in a file to use the latest and most secure protection method, by re-encrypting the passwords using the **mqiptPW** utility. For more information on encrypting passwords in the MQIPT configuration, see Encrypting stored passwords.

If you change **SSLClientKeyRingPW** (and **SSLClient** is set to `true`), the route is stopped, and restarted when a refresh command is issued. All connections to the route are stopped.

SSLClientKeyRingUseCryptoHardware

Specifies whether cryptographic hardware that supports the PKCS #11 interface is used as the key store containing the client certificate, when MQIPT is acting as a SSL/TLS client. If this property is set to `true`, **SSLClientKeyRing** cannot be set on the same route.

If you change this property (and **SSLClient** is set to `true`), the route is stopped, and restarted when a refresh command is issued. All connections to the route are stopped.

Use of cryptographic hardware with MQIPT is an IBM MQ Advanced capability. To use this capability, the local queue manager that is connected using the MQIPT route is also required to

have IBM MQ Advanced, IBM MQ Appliance, IBM MQ Advanced for z/OS, or IBM MQ Advanced for z/OS VUE entitlement. The route will not start when this property is set to `true` unless the **EnableAdvancedCapabilities** global property is set to confirm that IBM MQ Advanced capabilities can be used.

V 9.3.0 **SSLClientOutboundSNI**

Specifies the value of the Server Name Indication (SNI) extension when MQIPT initiates a TLS connection to the route destination. The SNI is either used by IBM MQ queue managers to present the correct certificate during the TLS handshake, or to route connections to the destination, depending on the configuration.

This property is only applicable to routes that are defined with `SSLClient=true`, and cannot be specified for routes defined with `HTTP=true`. If you change the value of this property, and **SSLClient** is set to `true`, the route is stopped and restarted when a refresh command is issued.

Attention: If the destination channel is configured with a certificate label on the channel object **CERTLABL** field, you must set the **CERTLABL** setting to the channel value. If a client is forwarded without the channel SNI setting, it is rejected with an MQRC_SSL_INITIALIZATION_ERROR return code and an AMQ9673 message printed in the remote queue manager error logs.

The value of the property can be one of the following values:

hostname
The SNI is set to the hostname of the route destination. Use this option if the route connects to a load balancer or router that uses the SNI to route requests. For example, the Red Hat® OpenShift® Container Platform Router uses the SNI to route requests to the IBM MQ queue manager.

If the route destination is a queue manager, connection requests receive the default certificate of the remote queue manager during the TLS handshake, and so per-channel certificates cannot be used.

If the route destination is specified using an IP address, and a reverse DNS lookup cannot be performed, the SNI is blank.

This is the default value.

channel
The SNI is set to the IBM MQ channel name. Use this option to allow per-channel certificates to be used by the destination queue manager, if connections received by the route do not contain the channel name in the SNI for one of the following reasons:

- The route is configured to accept connections that are not secured with TLS with either `SSLServer=false` or `SSLPlainConnections=true`.
- The application that connects to the route cannot set the SNI, or is configured to set the SNI to a value other than the IBM MQ channel name.

passthru
If the route is defined with `SSLServer=true`, the SNI on the outbound connection is set to the value of the SNI received on the inbound connection to the route. If the route is not configured to accept TLS connections, the SNI is set to the destination hostname.

custom
The SNI is set to the value specified in the **SSLClientCustomOutboundSNI** property. If the **SSLClientCustomOutboundSNI** property is not specified, the SNI is set as if the route is configured with `SSLClientOutboundSNI=hostname`.

none
The SNI is not set.

SSLClientProtocols
Used to restrict the set of enabled secure socket protocols that are used to make outbound connections to the destination for a route when **SSLClient** is set to `true`.

You can specify multiple values by separating them with commas. In versions earlier than IBM MQ 9.2.5, if you do not specify this property, the only protocol enabled by default is TLS 1.2. From IBM

MQ 9.2.5, if you do not specify this property, TLS 1.2 and TLS 1.3 are enabled by default. To enable protocols other than TLS 1.2 or TLS 1.3, you must specify the protocols to enable in this property, and also add support for the protocol in the Java runtime environment by following the procedure in Enabling deprecated protocols and CipherSuites. You can specify one or more of the following values.

Table 118. Permitted values for SSL/TLS protocols

Value	Protocol
SSLv3	SSL 3.0
TLSv1	TLS 1.0
TLSv1.1	TLS 1.1
TLSv1.2	TLS 1.2
V 9.3.0 TLSv1.3	TLS 1.3

Use the entry listed in the **Value** column in the route property. The corresponding entry in the **Protocol** column is for information only.

SSLClientSiteDN_C
Use this property to specify a country name to select a certificate to send to the SSL/TLS server. Certificate matching is not case sensitive. If you do not specify this property, certificates are accepted with any country name. If you change this property (and **SSLClient** is set to `true`), the route is stopped, and restarted when a refresh command is issued. All connections to the route are stopped.

SSLClientSiteDN_CN
Use this property to specify a common name to select a certificate to send to the SSL/TLS server. Certificate matching is not case sensitive. If you do not specify this property, certificates are accepted with any common name. If you change this property (and **SSLClient** is set to `true`), the route is stopped, and restarted when a refresh command is issued. All connections to the route are stopped.

SSLClientSiteDN_DC
Use this property to specify a domain component name to select a certificate to send to the SSL/TLS server. Certificate matching is not case sensitive. You can specify multiple DCs by separating them with commas. Each DC represents an element in a domain name, for example the domain name `example.ibm.com` is represented as `example,ibm,com` using commas to separate the multiple values. If you do not specify this property, certificates are accepted with any domain component name. If you change this property (and **SSLClient** is set to `true`), the route is stopped, and restarted when a refresh command is issued. All connections to the route are stopped.

SSLClientSiteDN_DNQ
Use this property to specify a domain qualifier to select a certificate to send to the SSL/TLS server. Certificate matching is not case sensitive. If you do not specify this property, certificates are accepted with any domain qualifier. If you change this property (and **SSLClient** is set to `true`), the route is stopped, and restarted when a refresh command is issued. All connections to the route are stopped.

SSLClientSiteDN_L
Use this property to specify a Location name to select a certificate to send to the SSL/TLS server. Certificate matching is not case sensitive. If you do not specify this property, certificates are accepted with any location name. If you change this property (and **SSLClient** is set to `true`), the route is stopped, and restarted when a refresh command is issued. All connections to the route are stopped.

SSLClientSiteDN_O
Use this property to specify an Organization name to select a certificate to send to the SSL/TLS server. Certificate matching is not case sensitive. If you do not specify this property, certificates are accepted with any organization name. If you change this property (and **SSLClient** is set to `true`), the route is stopped, and restarted when a refresh command is issued. All connections to the route are stopped.

SSLClientSiteDN_OU
Use this property to specify an Organizational Unit (OU) name to select a certificate to send to the SSL/TLS server. You can specify multiple OUs by separating them with commas. (Match a literal

comma by prefixing it with a backslash (\) character.) Certificate matching is not case sensitive. If you do not specify this property, certificates are accepted with any OU name. If you change this property (and **SSLClient** is set to `true`), the route is stopped and restarted when a refresh command is issued. All connections to this route are stopped.

SSLClientSiteDN_PC
Use this property to specify a postal code to select a certificate to send to the SSL/TLS server. Certificate matching is not case sensitive. If you do not specify this property, certificates are accepted with any postal code. If you change this property (and **SSLClient** is set to `true`), the route is stopped, and restarted when a refresh command is issued. All connections to the route are stopped.

SSLClientSiteDN_ST
Use this property to specify a State name to select a certificate to send to the SSL/TLS server. Certificate matching is not case sensitive. If you do not specify this property, certificates are accepted with any state name. If you change this property (and **SSLClient** is set to `true`), the route is stopped, and restarted when a refresh command is issued. All connections to the route are stopped.

SSLClientSiteDN_Street
Use this property to specify a street name to select a certificate to send to the SSL/TLS server. Certificate matching is not case sensitive. If you do not specify this property, certificates are accepted with any street name. If you change this property (and **SSLClient** is set to `true`), the route is stopped, and restarted when a refresh command is issued. All connections to the route are stopped.

SSLClientSiteDN_T
Use this property to specify a title to select a certificate to send to the SSL/TLS server. Certificate matching is not case sensitive. If you do not specify this property, certificates are accepted with any title. If you change this property (and **SSLClient** is set to `true`), the route is stopped, and restarted when a refresh command is issued. All connections to the route are stopped.

SSLClientSiteDN_UID
Use this property to specify a user ID to select a certificate to send to the SSL/TLS server. Certificate matching is not case sensitive. If you do not specify this property, certificates are accepted with any user ID. If you change this property (and **SSLClient** is set to `true`), the route is stopped, and restarted when a refresh command is issued. All connections to the route are stopped.

SSLClientSiteLabel
Use this property to specify a label name to select a certificate to send to the SSL/TLS server. If you do not specify this property, certificates are accepted with any label name. If you change this property (and **SSLClient** is set to `true`), the route is stopped, and restarted when a refresh command is issued. All connections to the route are stopped.

SSLExitData
Use this property to provide a user-defined string to be passed to the exit.

SSLExitName
Use this property to define the class name for the exit that will be called when the route is acting as an SSL/TLS client or an SSL/TLS server. The name must include any package name; for example, `com.ibm.mq.ipt.exit.TestExit`.

SSLExitPath
Use this property to define the location of the exit to be used to load a copy of the exit. The name must be a fully qualified name to be used to locate the class file or the name of a `.jar` file that contains the class file; for example, `C:\mqipt\exits` or `C:\mqipt\exits\exits.jar`.

SSLExitTimeout
Use this property to define how long MQIPT waits for the exit to complete before terminating the connection request. A value of `0` means that MQIPT waits indefinitely.

SSLPlainConnections
Use this property to specify whether SSL/TLS is mandatory for connections to the MQIPT listener port of a route configured to accept inbound SSL/TLS connections. This property is applicable to routes that have either the **SSLServer** or **SSLProxyMode** property set to true. If enabled, this property allows unencrypted connections to connect to the route listener port, which means that MQIPT can forward all IBM MQ connections to the queue manager's listener port regardless of whether the connection is encrypted. If you do not set this parameter, or set it to `false`, only inbound SSL/TLS

connections are allowed. If you change this property, the route is stopped, and restarted when a refresh command is issued. All connections to the route are stopped.

SSLProxyMode
Set this property to `true` to make the route accept only SSL/TLS client connection requests and to tunnel the request directly to the destination. If you change this property, the route is stopped and restarted when a refresh command is issued. All connections to this route are stopped. This property cannot be used in conjunction with the following properties:

- **SocksClient**
- **SocksServer**
- **SSLClient**
- **SSLServer**

SSLServer
Set this property to `true` to make the route act as an SSL/TLS server and accept incoming SSL/TLS connections. Setting **SSLServer** to `true` implies that the caller is another MQIPT acting as an SSL/TLS client, or is an IBM MQ client or queue manager with SSL/TLS enabled.

If you set **SSLServer** to `true`, you must specify a SSL/TLS server key ring using the **SSLServerKeyRing** property, or configure MQIPT to use cryptographic hardware by setting the **SSLServerKeyRingUseCryptoHardware** property.

If you change this property, the route is stopped, and restarted when a refresh command is issued. All connections to the route are stopped.

This property cannot be used in conjunction with the following properties:

- **SocksServer**
- **SSLProxyMode**

SSLServerCAKeyRing
The fully-qualified file name of the key ring file containing CA certificates, used to authenticate certificates from the SSL/TLS client. On Windows platforms, you must use a double backslash (\\) as the file separator. If you change this property (and **SSLServer** is set to `true`), the route is stopped, and restarted when a refresh command is issued. All connections to this route are stopped.

SSLServerCAKeyRingPW
The password to open the SSL/TLS server CA key ring file specified with the **SSLServerCAKeyRing** property, or to connect to the cryptographic hardware key store if the **SSLServerCAKeyRingUseCryptoHardware** property is set to `true`.

The value can be either a password that has been encrypted using the **mqiptPW** command, or the fully-qualified file name of the file containing an encrypted password. If you specify a file name on Windows platforms, you must use a double backslash (\\) as the file separator. You are encouraged to migrate any key ring passwords currently stored in a file to use the latest and most secure protection method, by re-encrypting the passwords using the **mqiptPW** utility. For more information on encrypting passwords in the MQIPT configuration, see Encrypting stored passwords.

If you change this property (and **SSLServer** is set to `true`), the route is stopped, and restarted when a refresh command is issued. All connections to the route are stopped.

SSLServerCAKeyRingUseCryptoHardware

Specifies whether cryptographic hardware that supports the PKCS #11 interface is used as the key store for the CA certificates, used to authenticate certificates from the SSL/TLS client. If this property is set to `true`, **SSLServerCAKeyRing** cannot be set on the same route.

If you change this property (and **SSLServer** is set to `true`), the route is stopped, and restarted when a refresh command is issued. All connections to the route are stopped.

Use of cryptographic hardware with MQIPT is an IBM MQ Advanced capability. To use this capability, the local queue manager that is connected using the MQIPT route is also required to have IBM MQ Advanced, IBM MQ Appliance, IBM MQ Advanced for z/OS, or IBM MQ Advanced

for z/OS VUE entitlement. The route will not start when this property is set to `true` unless the **EnableAdvancedCapabilities** global property is set to confirm that IBM MQ Advanced capabilities can be used.

SSLServerAskClientAuth
Use this property to request SSL/TLS client authentication by the SSL/TLS server. The SSL/TLS client must have its own certificate to send to the SSL/TLS server. The certificate is retrieved from the key ring file. If you change this property (and **SSLServer** is set to `true`), the route is stopped, and restarted when a refresh command is issued. All connections to this route are stopped.

SSLServerCipherSuites
The name of the SSL/TLS CipherSuite to use on the SSL/TLS server side. This can be one or more of the supported CipherSuites. If you leave this blank, any CipherSuite for the enabled protocols that is compatible with the server certificate in the key ring is used. If you change this property (and **SSLServer** is set to `true`), the route is stopped, and restarted when a refresh command is issued. All connections to this route are stopped.

SSLServerDN_C
Use this property to accept certificates received from the SSL/TLS client of this country name. The name can be prefixed or suffixed with an asterisk (*) to extend its scope. Certificate matching is not case sensitive. If you do not specify this property, certificates are accepted with any company name. If you change this property (and **SSLServer** is set to `true`), the route is stopped, and restarted when a refresh command is issued. All connections to the route are stopped.

SSLServerDN_CN
Use this property to accept certificates received from the SSL/TLS client of this common name. The name can be prefixed or suffixed with an asterisk (*) to extend its scope. Certificate matching is not case sensitive. If you do not specify this property, certificates are accepted with any common name. If you change this property (and **SSLServer** is set to `true`), the route is stopped, and restarted when a refresh command is issued. All connections to the route are stopped.

SSLServerDN_DC
Use this property to accept certificates received from the SSL/TLS client of this domain component name. The name can be prefixed or suffixed with an asterisk (*) to extend its scope. Certificate matching is not case sensitive. You can specify multiple DCs by separating them with commas. Each DC represents an element in a domain name, for example the domain name `example.ibm.com` is represented as `example,ibm,com` using commas to separate the multiple values. If you do not specify this property, certificates are accepted with any domain component name. If you change this property (and **SSLServer** is set to `true`), the route is stopped, and restarted when a refresh command is issued. All connections to the route are stopped.

SSLServerDN_DNQ
Use this property to accept certificates received from the SSL/TLS client of this domain qualifier. The name can be prefixed or suffixed with an asterisk (*) to extend its scope. Certificate matching is not case sensitive. If you do not specify this property, certificates are accepted with any domain qualifier. If you change this property (and **SSLServer** is set to `true`), the route is stopped, and restarted when a refresh command is issued. All connections to the route are stopped.

SSLServerDN_L
Use this property to accept certificates received from the SSL/TLS client of this location. The name can be prefixed or suffixed with an asterisk (*) to extend its scope. Certificate matching is not case sensitive. If you do not specify this property, certificates are accepted with any location. If you change this property (and **SSLServer** is set to `true`), the route is stopped, and restarted when a refresh command is issued. All connections to the route are stopped.

SSLServerDN_O
Use this property to accept certificates received from the SSL/TLS client of this organization. The name can be prefixed or suffixed with an asterisk (*) to extend its scope. Certificate matching is not case sensitive. If you do not specify this property, certificates are accepted with any organization. If you change this property (and **SSLServer** is set to `true`), the route is stopped, and restarted when a refresh command is issued. All connections to the route are stopped.

SSLServerDN_OU
Use this property to accept certificates received from the SSL/TLS client of this Organizational Unit (OU). The name can be prefixed or suffixed with an asterisk (*) to extend its scope. You can specify multiple OUs by separating them with commas. (Match a literal comma by prefixing it with a backslash (\) character.) Certificate matching is not case sensitive. If you do not specify this property, certificates are accepted with any OU name. If you change this property (and **SSLServer** is set to `true`), the route is stopped and restarted when a refresh command is issued. All connections to this route are stopped.

SSLServerDN_PC
Use this property to accept certificates received from the SSL/TLS client of this postal code. The name can be prefixed or suffixed with an asterisk (*) to extend its scope. Certificate matching is not case sensitive. If you do not specify this property, certificates are accepted with any postal code. If you change this property (and **SSLServer** is set to `true`), the route is stopped, and restarted when a refresh command is issued. All connections to the route are stopped.

SSLServerDN_ST
Use this property to accept certificates received from the SSL/TLS client of this state. The name can be prefixed or suffixed with an asterisk (*) to extend its scope. Certificate matching is not case sensitive. If you do not specify this property, certificates are accepted with any state. If you change this property (and **SSLServer** is set to `true`), the route is stopped, and restarted when a refresh command is issued. All connections to the route are stopped.

SSLServerDN_Street
Use this property to accept certificates received from the SSL/TLS client of this street name. The name can be prefixed or suffixed with an asterisk (*) to extend its scope. Certificate matching is not case sensitive. If you do not specify this property, certificates are accepted with any street name. If you change this property (and **SSLServer** is set to `true`), the route is stopped, and restarted when a refresh command is issued. All connections to the route are stopped.

SSLServerDN_T
Use this property to accept certificates received from the SSL/TLS client of this title. The name can be prefixed or suffixed with an asterisk (*) to extend its scope. Certificate matching is not case sensitive. If you do not specify this property, certificates are accepted with any title. If you change this property (and **SSLServer** is set to `true`), the route is stopped, and restarted when a refresh command is issued. All connections to the route are stopped.

SSLServerDN_UID
Use this property to accept certificates received from the SSL/TLS client of this user ID. The name can be prefixed or suffixed with an asterisk (*) to extend its scope. Certificate matching is not case sensitive. If you do not specify this property, certificates are accepted with any user ID. If you change this property (and **SSLServer** is set to `true`), the route is stopped, and restarted when a refresh command is issued. All connections to the route are stopped.

SSLServerExit
Use this property to enable or disable the use of an exit when the route is acting as an SSL/TLS server. This allows you to define exit details in the configuration file without them actually being used.

SSLServerKeyRing
The fully-qualified file name of the key ring file containing the server certificate. On Windows platforms, you must use a double backslash (\\) as the file separator. If you change this property (and **SSLServer** is set to `true`), the route is stopped, and restarted when a refresh command is issued. All connections to the route are stopped.

SSLServerKeyRingPW
The password to open the SSL/TLS server key ring file specified with the **SSLServerKeyRing** property, or to connect to the cryptographic hardware key store if the **SSLServerKeyRingUseCryptoHardware** property is set to `true`.

The value can be either a password that has been encrypted using the **mqiptPW** command, or the fully-qualified file name of the file containing an encrypted password. If you specify a file name on Windows platforms, you must use a double backslash (\\) as the file separator. You are encouraged to migrate any key ring passwords currently stored in a file to use the latest and most secure

protection method, by re-encrypting the passwords using the **mqiptPW** utility. For more information on encrypting passwords in the MQIPT configuration, see Encrypting stored passwords.

You must specify **SSLServerKeyRingPW** if you set **SSLServer** to `true`.

If you change this property (and **SSLServer** is set to `true`), the route is stopped, and restarted when a refresh command is issued. All connections to the route are stopped.

SSLServerKeyRingUseCryptoHardware

Specifies whether cryptographic hardware that supports the PKCS #11 interface is used as the key store for the server certificate, when MQIPT is acting as a SSL/TLS server. If this property is set to `true`, **SSLServerKeyRing** cannot be set on the same route.

If you change this property (and **SSLServer** is set to `true`), the route is stopped, and restarted when a refresh command is issued. All connections to the route are stopped.

Use of cryptographic hardware with MQIPT is an IBM MQ Advanced capability. To use this capability, the local queue manager that is connected using the MQIPT route is also required to have IBM MQ Advanced, IBM MQ Appliance, IBM MQ Advanced for z/OS, or IBM MQ Advanced for z/OS VUE entitlement. The route will not start when this property is set to `true` unless the **EnableAdvancedCapabilities** global property is set to confirm that IBM MQ Advanced capabilities can be used.

SSLServerProtocols

Used to restrict the set of enabled secure socket protocols that are used to accept inbound connections to the route listener port for a route when **SSLServer** is set to `true`).

You can specify multiple values by separating them with commas. In versions earlier than IBM MQ 9.2.5, if you do not specify this property, the only protocol enabled by default is TLS 1.2. From IBM MQ 9.2.5, if you do not specify this property, TLS 1.2 and TLS 1.3 are enabled by default. To enable protocols other than TLS 1.2 or TLS 1.3, you must specify the protocols to enable in this property, and also add support for the protocol in the Java runtime environment by following the procedure in Enabling deprecated protocols and CipherSuites. You can specify one or more of the following values.

Table 119. Permitted values for SSL/TLS protocols

Value	Protocol
SSLv3	SSL 3.0
TLSv1	TLS 1.0
TLSv1.1	TLS 1.1
TLSv1.2	TLS 1.2
V 9.3.0 TLSv1.3	TLS 1.3

Use the entry listed in the **Value** column in the route property. The corresponding entry in the **Protocol** column is for information only.

SSLServerSiteDN_C

Use this property to specify a country name to select a certificate to send to the SSL/TLS client. Certificate matching is not case sensitive. If you do not specify this property, certificates are accepted with any country name. If you change this property (and **SSLServer** is set to `true`), the route is stopped, and restarted when a refresh command is issued. All connections to the route are stopped.

SSLServerSiteDN_CN

Use this property to specify a Common Name to select a certificate to send to the SSL/TLS client. Certificate matching is not case sensitive. If you do not specify this property, certificates are accepted with any common name. If you change this property (and **SSLServer** is set to `true`), the route is stopped, and restarted when a refresh command is issued. All connections to the route are stopped.

SSLServerSiteDN_DC
Use this property to specify a domain component name to select a certificate to send to the SSL/TLS client. Certificate matching is not case sensitive. You can specify multiple DCs by separating them with commas. Each DC represents an element in a domain name, for example the domain name `example.ibm.com` is represented as `example,ibm,com` using commas to separate the multiple values. If you do not specify this property, certificates are accepted with any domain component name. If you change this property (and **SSLServer** is set to `true`), the route is stopped, and restarted when a refresh command is issued. All connections to the route are stopped.

SSLServerSiteDN_DNQ
Use this property to specify a domain qualifier to select a certificate to send to the SSL/TLS client. Certificate matching is not case sensitive. If you do not specify this property, certificates are accepted with any domain qualifier. If you change this property (and **SSLServer** is set to `true`), the route is stopped, and restarted when a refresh command is issued. All connections to the route are stopped.

SSLServerSiteDN_L
Use this property to specify a Location name to select a certificate to send to the SSL/TLS client. Certificate matching is not case sensitive. If you do not specify this property, certificates are accepted with any location name. If you change this property (and **SSLServer** is set to `true`), the route is stopped, and restarted when a refresh command is issued. All connections to the route are stopped.

SSLServerSiteDN_O
Use this property to specify an organization name to select a certificate to send to the SSL/TLS client. Certificate matching is not case sensitive. If you do not specify this property, certificates are accepted with any organization name. If you change this property (and **SSLServer** is set to `true`), the route is stopped, and restarted when a refresh command is issued. All connections to the route are stopped.

SSLServerSiteDN_OU
Use this property to specify an Organizational Unit (OU) name to select a certificate to send to the SSL/TLS client. You can specify multiple OUs by separating them with commas. (Match a literal comma by prefixing it with a backslash (\) character.) Certificate matching is not case sensitive. If you do not specify this property, certificates are accepted with any OU name. If you change this property (and **SSLServer** is set to `true`), the route is stopped and restarted when a refresh command is issued. All connections to this route are stopped.

SSLServerSiteDN_PC
Use this property to specify a postal code to select a certificate to send to the SSL/TLS client. Certificate matching is not case sensitive. If you do not specify this property, certificates are accepted with any postal code. If you change this property (and **SSLServer** is set to `true`), the route is stopped, and restarted when a refresh command is issued. All connections to the route are stopped.

SSLServerSiteDN_ST
Use this property to specify a State name to select a certificate to send to the SSL/TLS client. Certificate matching is not case sensitive. If you do not specify this property, certificates are accepted with any state name. If you change this property (and **SSLServer** is set to `true`), the route is stopped, and restarted when a refresh command is issued. All connections to the route are stopped.

SSLServerSiteDN_Street
Use this property to specify a street name to select a certificate to send to the SSL/TLS client. Certificate matching is not case sensitive. If you do not specify this property, certificates are accepted with any street name. If you change this property (and **SSLServer** is set to `true`), the route is stopped, and restarted when a refresh command is issued. All connections to the route are stopped.

SSLServerSiteDN_T
Use this property to specify a title to select a certificate to send to the SSL/TLS client. Certificate matching is not case sensitive. If you do not specify this property, certificates are accepted with any title. If you change this property (and **SSLServer** is set to `true`), the route is stopped, and restarted when a refresh command is issued. All connections to the route are stopped.

SSLServerSiteDN_UID
Use this property to specify a user ID to select a certificate to send to the SSL/TLS client. Certificate matching is not case sensitive. If you do not specify this property, certificates are accepted with any user ID. If you change this property (and **SSLServer** is set to `true`), the route is stopped, and restarted when a refresh command is issued. All connections to the route are stopped.

SSLServerSiteLabel
Use this property to specify a label name to select a certificate to send to the SSL/TLS client. If you do not specify this property, certificates are accepted with any label name. If you change this property (and **SSLServer** is set to `true`), the route is stopped, and restarted when a refresh command is issued. All connections to the route are stopped.

StoredCredentialsFormat
Use this property to indicate whether the values of password properties use the encrypted password format supported in MQIPT from IBM MQ 9.1.5. MQIPT can almost always detect whether passwords are specified in the encrypted password format. This property only needs to be set in the unlikely scenario that MQIPT cannot automatically differentiate between an encrypted password and a plain text password or a file name.

The value can be one of the following values:

encrypted
Password properties contain an encrypted password in the format that is supported in MQIPT from IBM MQ 9.1.5.

compat
Password properties contain either a plain text password, or for key ring passwords, the name of the file containing an encrypted password.

TCPKeepAlive
Set this property to `true` to enable the sending of TCP/IP keep-alive packets periodically to prevent the connections on this route becoming idle. This reduces the chances of the MQIPT connections being severed by a firewall or router. The sending of TCP/IP keep-alive packets is controlled by operating system tuning parameters; consult your operating system documentation for further details on how to tune keep-alive. If you do not set this parameter, or set it to `false`, keep-alive packets are not sent.

Trace
The level of tracing required for this route. Enabling trace for one route does not enable trace for any other routes. If you need to trace more than one route, you must add the **Trace** property to the `[route]` section of each route to be traced.

The value of this property can be one of the following:

0
Trace is not enabled

Any positive integer
Trace is enabled

The default value is `0`.

If the `[route]` section does not include a **Trace** property, the **Trace** property from the `[global]` section is used. For information about tracing threads that are not associated with a route, see **Trace** in the `[global]` section. If a change to this property affects a route, the new value is used when the refresh command is issued. All connections use the new value immediately. The route is not stopped.

TraceUserData
The amount of user data in network transmissions received and sent by this route that is traced, when trace is enabled for this route. The value can be one of the following values:

0
No user data is traced.

all
All user data is traced.

numberOfBytes
The specified number of bytes of data, including the transmission segment header (TSH), is traced. The value specified must be greater than 15.

UriName

This property can be used to change the name of the Uniform Resource Identifier of the resource when using an HTTP proxy, although the default value will suffice for most configurations:

```
HTTP://destination:destination_port/mqipt
```

If you change this property (and **HTTP** is set to `true`), the route is stopped, and restarted when a refresh command is issued.

mqiptAdmin properties

The **mqiptAdmin** command reads configuration properties from a properties file that is specified when the command is started.

The following properties can be specified in the properties file that is used by the **mqiptAdmin** command. Property names are case-sensitive.

PasswordProtectionKeyFile
The name of the file containing the encryption key used to encrypt the trust store password that is specified in the **SSLClientCAKeyRingPW** property. If this property is not specified, the default encryption key is used to decrypt the password. The encryption key used to encrypt the **mqiptAdmin** trust store password can be different to the encryption key used to encrypt passwords in the `mqipt.conf` configuration file.

SSLClientCAKeyRing
The file name of the PKCS#12 trust store to use for connections to the MQIPT TLS command port. The trust store should contain the CA certificate of the CA that signed the server certificate that the MQIPT TLS command port is configured to use. Backslash (\) characters in the file name must be escaped and specified as a double backslash (\\).

SSLClientCAKeyRingPW
The encrypted password to access the trust store specified using the **SSLClientCAKeyRing** property. The password must be encrypted using the **mqiptPW** command, and the value of this property set to the string output by **mqiptPW**.

Notices

This information was developed for products and services offered in the U.S.A.

IBM may not offer the products, services, or features discussed in this document in other countries. Consult your local IBM representative for information on the products and services currently available in your area. Any reference to an IBM product, program, or service is not intended to state or imply that only that IBM product, program, or service may be used. Any functionally equivalent product, program, or service that does not infringe any IBM intellectual property right may be used instead. However, it is the user's responsibility to evaluate and verify the operation of any non-IBM product, program, or service.

IBM may have patents or pending patent applications covering subject matter described in this document. The furnishing of this document does not grant you any license to these patents. You can send license inquiries, in writing, to:

IBM Director of Licensing
IBM Corporation
North Castle Drive
Armonk, NY 10504-1785
U.S.A.

For license inquiries regarding double-byte (DBCS) information, contact the IBM Intellectual Property Department in your country or send inquiries, in writing, to:

Intellectual Property Licensing
Legal and Intellectual Property Law
IBM Japan, Ltd.
19-21, Nihonbashi-Hakozakicho, Chuo-ku
Tokyo 103-8510, Japan

The following paragraph does not apply to the United Kingdom or any other country where such provisions are inconsistent with local law: INTERNATIONAL BUSINESS MACHINES CORPORATION PROVIDES THIS PUBLICATION "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. Some states do not allow disclaimer of express or implied warranties in certain transactions, therefore, this statement may not apply to you.

This information could include technical inaccuracies or typographical errors. Changes are periodically made to the information herein; these changes will be incorporated in new editions of the publication. IBM may make improvements and/or changes in the product(s) and/or the program(s) described in this publication at any time without notice.

Any references in this information to non-IBM Web sites are provided for convenience only and do not in any manner serve as an endorsement of those Web sites. The materials at those Web sites are not part of the materials for this IBM product and use of those Web sites is at your own risk.

IBM may use or distribute any of the information you supply in any way it believes appropriate without incurring any obligation to you.

Licensees of this program who wish to have information about it for the purpose of enabling: (i) the exchange of information between independently created programs and other programs (including this one) and (ii) the mutual use of the information which has been exchanged, should contact:

IBM Corporation
Software Interoperability Coordinator, Department 49XA
3605 Highway 52 N
Rochester, MN 55901
U.S.A.

Such information may be available, subject to appropriate terms and conditions, including in some cases, payment of a fee.

The licensed program described in this information and all licensed material available for it are provided by IBM under terms of the IBM Customer Agreement, IBM International Program License Agreement, or any equivalent agreement between us.

Any performance data contained herein was determined in a controlled environment. Therefore, the results obtained in other operating environments may vary significantly. Some measurements may have been made on development-level systems and there is no guarantee that these measurements will be the same on generally available systems. Furthermore, some measurements may have been estimated through extrapolation. Actual results may vary. Users of this document should verify the applicable data for their specific environment.

Information concerning non-IBM products was obtained from the suppliers of those products, their published announcements or other publicly available sources. IBM has not tested those products and cannot confirm the accuracy of performance, compatibility or any other claims related to non-IBM products. Questions on the capabilities of non-IBM products should be addressed to the suppliers of those products.

All statements regarding IBM's future direction or intent are subject to change or withdrawal without notice, and represent goals and objectives only.

This information contains examples of data and reports used in daily business operations. To illustrate them as completely as possible, the examples include the names of individuals, companies, brands, and products. All of these names are fictitious and any similarity to the names and addresses used by an actual business enterprise is entirely coincidental.

COPYRIGHT LICENSE:

This information contains sample application programs in source language, which illustrate programming techniques on various operating platforms. You may copy, modify, and distribute these sample programs in any form without payment to IBM, for the purposes of developing, using, marketing or distributing application programs conforming to the application programming interface for the operating platform for which the sample programs are written. These examples have not been thoroughly tested under all conditions. IBM, therefore, cannot guarantee or imply reliability, serviceability, or function of these programs.

If you are viewing this information softcopy, the photographs and color illustrations may not appear.

Programming interface information

Programming interface information, if provided, is intended to help you create application software for use with this program.

This book contains information on intended programming interfaces that allow the customer to write programs to obtain the services of WebSphere MQ.

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

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

Trademarks

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

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

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

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

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

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

Part Number:

(1P) P/N: