

9.4

IBM MQ Scenarios



Note

Before using this information and the product it supports, read the information in [“Notices” on page 215.](#)

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

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

© **Copyright International Business Machines Corporation 2007, 2024.**

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

Contents

Scenarios.....	5
Getting started with IBM MQ.....	5
Planning the solution.....	5
Implementing the solution.....	7
What to do next.....	18
Point-to-point scenario.....	18
Planning the solution.....	18
Implementing the solution.....	20
Securing the point-to-point topology.....	26
Streaming queues.....	29
Streaming queues configuration.....	30
Streaming to remote and alias queues.....	31
Streaming queue restrictions.....	32
Stream queues and transactions.....	33
Streaming to and from cluster queues.....	33
Using streaming queues to store a history of messages.....	34
Publish/subscribe scenarios.....	35
Scenario: Creating a publish/subscribe cluster.....	35
Publish/subscribe hierarchy scenarios.....	41
Transactional support scenarios.....	51
Introducing units of work.....	51
Scenario 1: Queue manager performs the coordination.....	52
Scenario 2: Other software provides the coordination.....	76
Expiring global units of work.....	83
Unit of recovery disposition.....	84
Security scenarios.....	85
Security scenario: two queue managers on z/OS.....	85
Security scenario: queue sharing group on z/OS.....	92
Server-to-server message channel interception example configurations.....	97
Connecting two queue managers using SSL/TLS.....	99
Connecting a client to a queue manager securely.....	106
Migrating on Windows.....	112
Planning the solution.....	112
Implementing the solution using the graphical user interface.....	116
Installing a later version of IBM MQ to coexist with an earlier version on Windows.....	141
Overview of multiple installations.....	141
Installing a later version of IBM MQ side-by-side to an earlier version.....	142
Using the setmqenv command to run with both versions of IBM MQ.....	144
Creating a queue manager.....	146
Migrating a queue manager to a later version of IBM MQ	147
Installing a fix pack on IBM MQ 9.4.....	149
Managed File Transfer scenario.....	151
MFT common topologies.....	151
Configuring the base server.....	154
Getting started with IBM MQ Internet Pass-Thru.....	162
Verifying that MQIPT is working correctly.....	163
Creating test certificates.....	165
Request a CA-signed certificate for MQIPT.....	166
Authenticating a TLS server.....	168
Authenticating a TLS client.....	170
Authenticating a TLS client and server.....	173
Configuring HTTP tunneling.....	176

Configuring access control.....	178
Configuring a SOCKS proxy.....	180
Configuring a SOCKS client.....	182
Configuring MQIPT clustering support.....	184
Allocating port numbers.....	187
Retrieving CRLs by using an LDAP server.....	188
Running MQIPT in TLS proxy mode.....	191
Running MQIPT in TLS proxy mode with a security manager.....	192
Using a security exit.....	195
Routing client connection requests to IBM MQ queue manager servers by using security exits....	197
Dynamically routing client connection requests.....	200
Using a certificate exit to authenticate a TLS server.....	203
Kafka Connect scenarios.....	205
Kafka Connect common topologies.....	206
Exactly once support.....	214
Notices.....	215
Programming interface information.....	216
Trademarks.....	216

IBM MQ Scenarios

Each scenario walks you through a significant set of tasks, and helps you to configure a major product feature. The scenarios include useful links to other content to help you to gain a better understanding of the area in which you are interested.

The available IBM MQ scenarios are described in the following subtopics.

Windows Getting started with IBM MQ

This scenario explains how to get started with IBM MQ on a Windows platform. Use this scenario if you have never used IBM MQ and want to get started quickly.

This scenario describes the basic steps for installing, configuring and verifying IBM MQ on Windows if you do not already have it installed on your system. You can complete the steps of the scenario by using either the graphical user interface or command-line interface.

Planning the solution

Choose a method for installing IBM MQ on Windows. Use the graphical user interface and wizards that take you through the installation and configuring process or use the command line to conduct a silent installation.

Overview: The delivered logical topology

The delivered logical topology after completing the scenario.

The installed IBM MQ server instance allows for creation of IBM MQ objects: queues and queue managers. You can use IBM MQ Explorer to put and get messages from the local queue through the queue manager. After this scenario is complete, the delivered topology will look like [Figure 1](#).

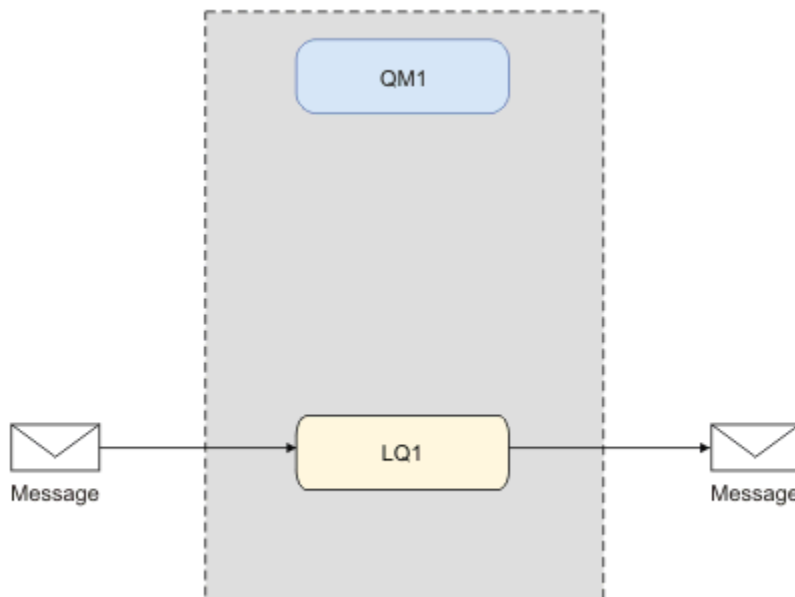


Figure 1. Put message on LQ1, get message from LQ1.

Basic concepts and key terms

Description of the basic concepts and key terms you must know about before using the Getting started with IBM MQ scenario.

Basic concepts

IBM MQ enables applications to read and write messages to a queue. The application that reads the message is independent of the application that writes the message. It is not a requirement to have the two applications running at the same time. If no application is available to read the message it is queued on the IBM MQ queue until an application reads it.

In this scenario you can choose to install and configure IBM MQ in one of the following ways:

“Installing and configuring using the graphical user interface” on page 7

During installation using the graphical user interface, you are guided through several screens and wizards to help you apply the relevant options and settings:

Launchpad

Check software requirements, specify network information and start the IBM MQ installation wizard.

IBM MQ installation wizard

Install the software and start the Prepare IBM MQ Wizard.

Prepare IBM MQ Wizard

Start the IBM MQ service and IBM MQ Explorer.

IBM MQ Explorer

Manage queues and queue managers.

“Installing and configuring using the command line interface” on page 11

The command line interface installation can be silent or interactive. The silent installation is fully accessible and is the one covered in this scenario. During installation using the command line, you are guided through several steps to help you apply relevant options and settings:

- Install IBM MQ
- Create and configure IBM MQ objects; queue managers and queues.
- Verify the installation by using `amqspu` to put and `amqsget` to get a message from the queue.

As well as using IBM MQ Explorer and command line to create IBM MQ objects, it is possible to do so by using the programmable interface. This is not included in the current scenario.

Key terms

Here is a list of key terms about message queuing.

Key terms about message queuing.

Term	Description
Queue managers	The queue manager is responsible for maintaining the queues it owns, and for storing all the messages it receives onto the appropriate queues.
Messages	A message is a string of bytes that is meaningful to the applications that use it. Messages are used to transfer information from one application program to another. The applications can be running on the same or on different computers.
Local queues	A local queue is a data structure used to store messages. The queue can be a normal queue or a transmission queue. A normal queue holds messages that are to be read by an application that is reading the message directly from the queue manager. A transmission queue holds messages that are in transit to another queue manager.

Implementing the solution

Implement the solution to the scenario. Install IBM MQ on Windows using the installation Launchpad, then verify the installation using the IBM MQ Explorer.

Installing and configuring using the graphical user interface

Install IBM MQ on Windows by using the installation launchpad and verify your installation by using IBM MQ Explorer. After verifying your installation, create a queue manager and a queue, and then try putting a message to the queue and getting a message from the queue.

Installing using the launchpad

Install IBM MQ on Windows using the installation launchpad.

Before you begin

Before starting this task, complete the following checks:

- You must have local administrator authority when you are installing. Define this authority through the Windows facilities.
- Ensure that the machine name does not contain any spaces.
- Ensure that you have sufficient disk space to fully install IBM MQ for Windows. For more information, see [Disk space requirements on Multiplatforms](#).
- Determine whether you need to define Windows domain user ID's for any IBM MQ users.

Before you install IBM MQ, check that your system meets the hardware and software requirements. For more information about hardware and software requirements on all supported platforms, see [System Requirements for IBM MQ](#).

About this task

The launchpad and subsequent wizards take you through the installation process, and help you review the software requirements and IBM MQ settings.

This task assumes that you are installing IBM MQ for the first time on your machine, and that you will be using the default locations. By default the location of the IBM MQ program files are C:\Program Files\IBM\MQ, and the data and log file location is C:\ProgramData\IBM\MQ.

Note: If you are installing IBM MQ 9.0, and you have any previous installations of IBM MQ on your machine, the location of the program and data files will be different from the default. For further information, see [Program and data directory locations](#). If you have already previously completed this scenario, and want to repeat it with a single, fresh installation using the default locations, remove your previous installation before starting the scenario again. To uninstall an existing instance of IBM MQ from your machine, see [“Uninstalling IBM MQ” on page 17](#).

The installation programs contain links to further information if you require it during the installation process.

Procedure

1. Start the Launchpad, review, and if necessary, modify the software requirements and network configuration.
 - a) Navigate to the IBM MQ software directory, and double-click the file Setup.exe to start the Launchpad.
 - b) Select the **Software Requirements** tab to display the **Software Requirements** settings.
 - c) Check that the software requirements have been met and that the entry for the requirement displays a green tick with the words OK. Make any indicated corrections.

Note:

For details of any requirement, click the check box to expand an information tab.

d) Select the **Network Configuration** tab to display the **Network Configuration** settings.

e) Select **No**.

Note: This scenario assumes that you do not need to configure a domain user ID for IBM MQ. For more information regarding configuring IBM MQ for Windows domain users, click **More information**.

f) On the **IBM MQ Installation** tab of the Launchpad, select the installation language, and then click **Launch IBM MQ Installer** to start the IBM MQ installation wizard.

You have completed setting up IBM MQ by meeting or specifying your installation requirements, and have started the IBM MQ installation wizard.

2. Use the IBM MQ installation wizard to install the software, and to start the Prepare IBM MQ Wizard.

a) In the IBM MQ installation wizard, read the License Agreement and click the **I accept the terms in the license agreement** check box, and then click **Next**.

b) Click **Typical**, and then click **Next**.

c) In the **Ready to Install IBM MQ** page, review the installation information and click **Install**.

Note: Note the following details:

- Installation Name
- Top-level folder for Program Files
- Top level folder for Data Files

The following features are installed:

- IBM MQ Server
- IBM MQ: a graphical interface for administering and monitoring IBM MQ resources
- Java and .NET Messaging and Web Services
- IBM MQ Development Toolkit

The installation process begins. Depending on your system the installation process can take several minutes.

At the end of the installation process, the IBM MQ Setup window displays the message **Installation Wizard Completed Successfully**.

d) Click **Finish**.

You have successfully installed IBM MQ. The Prepare IBM MQ Wizard starts automatically, displaying the **Prepare IBM MQ Wizard** page.

3. Use the Prepare IBM MQ Wizard to start the IBM MQ service.

a) On the Welcome to the Prepare IBM MQ Wizard page, select **Next**.

The Prepare IBM MQ Wizard displays the message **Status: Checking IBM MQ Configuration** and a progress bar. When the process is complete the IBM MQ Network Configuration page is displayed.

b) On the IBM MQ Network Configuration page of the Prepare IBM MQ Wizard, select **No**.

c) Click **Next**.

The Prepare IBM MQ Wizard displays a message **Status: starting the IBM MQ Service**, and a progress bar. When the process is complete the wizard displays the **Completing the Prepare IBM MQ Wizard** page.

d) Select **Launch IBM MQ Explorer** and choose whether to view the release notes, then click the **Finish** button.

IBM MQ Explorer starts.

You have installed IBM MQ. You have also started the IBM MQ Explorer.

Results

IBM MQ is installed and verified, and you are ready to configure objects such as queue managers and queues.

What to do next

Follow the instructions in [“Creating a queue manager called QM1” on page 9.](#)

Related concepts

[Disc space requirements](#)

[Hardware and software requirements on Windows systems](#)

[Introduction to IBM MQ](#)

Related tasks

[Installing an IBM MQ server on Windows](#)

[Configuring an IBM MQ server](#)

Creating a queue manager called QM1

Create a queue manager, called QM1 by using IBM MQ Explorer. Queue managers are the main components in an IBM MQ messaging network.

Before you begin

You must have IBM MQ installed. If you do not, see [“Installing using the launchpad” on page 7](#) for information about how to do so.

About this task

In this example, all names are typed in uppercase and because IBM MQ names are case-sensitive, you must type all names in uppercase too.

To create and start a queue manager by using IBM MQ Explorer, complete the following steps.

Procedure

1. Start IBM MQ Explorer as an administrator.
2. In the **Navigator** view, right-click the **Queue Managers** folder, then click **New > Queue Manager**. The **Create Queue Manager** wizard starts.
3. In the **Queue Manager name** field, type QM1.
4. Select the **Make this the default queue manager** check box.
5. In the **Dead-letter queue** field type SYSTEM.DEAD.LETTER.QUEUE.
This is the name of the dead-letter queue that is automatically created when you create the queue manager.
6. Leave the other fields empty and click **Finish**, or if that button is disabled, click **Next**.
The **Finish** button is disabled if the port number conflicts with an existing queue manager, for example the queue manager that is created as part of the default configuration. You must continue through the wizard to change the default port number.
7. If you clicked **Next**, continue to accept the defaults and click **Next** on each page until you get to the final page of the wizard, when the **Finish** button becomes available. Change the specified port number, for example to 1415, and click **Finish**.

IBM MQ displays a **Creating Queue Manager** dialog window while the queue manager is created and started.

What to do next

To create a queue, see [“Creating a queue called LQ1” on page 10.](#)

Related tasks

[Creating and managing queue managers on Multiplatforms](#)

Creating a queue called LQ1

Create a queue by using IBM MQ Explorer. Queues are data structures that are used to store messages and are IBM MQ queue manager objects.

About this task

In this task you can create IBM MQ objects using IBM MQ Explorer.

To create and start a queue by using IBM MQ Explorer, complete the following steps.

Procedure

1. In the **Navigator** view, expand the **Queue Managers** folder.
2. Expand queue manager **QM1**.
3. Right-click the **Queues** folder, then click **New > Local Queue...** The **New Local Queue** wizard starts.
4. In the **Name** field, type LQ1.
5. Click **Finish**.

The new queue LQ1, is displayed in the **Content** view. If the queue is not displayed in the **Content** view, click on the **Refresh** button, at the top of the **Content** view.

What to do next

You are ready to put a message on to your queue. To put a message in a queue, see [“Putting a message to the queue LQ1” on page 10](#).

Putting a message to the queue LQ1

Put a message on to the queue LQ1 by using IBM MQ Explorer.

About this task

This task assumes that you have already created a queue manager called QM1 as described in [“Creating a queue manager called QM1” on page 13](#) and a queue called LQ1 as described in [“Creating a queue called LQ1” on page 10](#).

To put a message to the queue by using IBM MQ Explorer, complete the following steps.

Procedure

1. In the **Navigator** view, expand the **Queue Managers** folder.
2. Expand queue manager QM1, which you created.
3. Click the **Queues** folder. The queue manager's queues are listed in the Content view.
4. In the Content view, right-click the local queue LQ1, then click **Put Test Message...**
The **Put test message** dialog opens.
5. In the **Message data** field, type some text, for example Hello World, then click **Put message**.
The **Message data** field is cleared and the message is put on the queue.
6. Click **Close**.

In the Content view, notice that the LQ1 **Current queue depth** value is now 1. If the **Current queue depth** column is not visible, you might need to scroll to the right of the **Content View**.

What to do next

To get a message from the queue, see [“Getting a message from the queue LQ1” on page 11](#).

Getting a message from the queue LQ1

Get a message from the queue LQ1 by using IBM MQ Explorer.

About this task

This task assumes that you have already put a message QM1 as described in [“Putting a message to the queue LQ1”](#) on page 10.

To get a message from the queue by using IBM MQ Explorer, complete the following steps.

Procedure

1. In the **Navigator** view, expand the **Queue Managers** folder, then expand QM1.
2. Click the **Queues** folder.
3. In the **Content** view, right-click the local queue LQ1, then click **Browse Messages....** The **Message browser** opens to show the list of the messages that are currently on QM1.
4. Double-click the last message to open the properties dialog.

On the **Data** page of the properties dialog, the **Message data** field displays the content of the message in human-readable form.

What to do next

Follow the instructions in subsequent scenarios to explore further IBM MQ features.

To learn about writing queuing applications, connecting to and disconnecting from a queue manager, publish/subscribe, and opening and closing objects, see [Writing a procedural application for queuing](#).

Installing and configuring using the command line interface

Install IBM MQ on Windows by using the command line to perform a silent installation and set up the environment variable. After verifying your installation, create a queue manager and a queue and then try putting a message to the queue and getting a message from the queue.

Installing using a silent installation

Install IBM MQ on Windows by using the command line to perform a silent installation and confirm that the environment for your installation is set up correctly.

Before you begin

Before you start this task, complete the following checks:

- You must have local administrator authority when you are installing. Define this authority through the Windows facilities.
- Ensure that the machine name does not contain any spaces.
- Ensure that you have sufficient disk space. For more information, see [Disk space requirements on Multiplatforms](#).
- Determine whether you need to define Windows domain user IDs for any IBM MQ users.

Before you install IBM MQ, check that your system meets the hardware and software requirements. For the latest details of hardware and software requirements on all supported platforms, see [System Requirements for IBM MQ](#).

About this task

This scenario assumes that you are installing IBM MQ for the first time on your machine, and that you are using the default locations. By default the location of the IBM MQ 9.0 program files are C:\Program Files\IBM\MQ, and the data and log file location is C:\ProgramData\IBM\MQ.

Note: If you have any previous installations of IBM MQ on your machine the default locations of the program and data files might change. For further information, see [Program and data directory locations](#). If you have already previously completed this scenario, and want to repeat it with a single, fresh installation using the default locations, remove your previous installation before starting the scenario again. To uninstall an existing instance of IBM MQ from your machine, see [“Uninstalling IBM MQ” on page 17](#).

IBM MQ on Windows uses the MSI technology to install software. For more information on installing using the MSI technology, see [Advanced installation using msixexec](#).

To install IBM MQ using the command line, you must specify the following parameters:

- `/i "MQ_INSTALLATION_MEDIA\MSI\IBM MQ.msi"` where `MQ_INSTALLATION_MEDIA` is the location of the `IBM MQ.msi` file. This argument specifies the location of the `.msi` file.
- `/l*v USER_LOGFILE_LOCATION\install.log` where `USER_LOGFILE_LOCATION` is where you want the installation logs to be written to.

Note: The folder where you want the `install.log` to be created must exist before you run the command.

- `/q[n|b|t|f]` `/q` must be paired with one of `n`, `b`, `t`, or `f`. Running the `msiexec` command at a command prompt opens the help file, which shows the correct usage.
- `USEINI="RESPONSE_FILE"` where `RESPONSE_FILE` is the name and location of the response file to be used by the silent installation. This scenario uses the sample `Response.ini` file, which is included in the IBM MQ installation media.
- `TRANSFORMS="TRANSFORM_FILE"` where `TRANSFORM_FILE` is the name of the transform file to be applied to the installation. This scenario uses the American English transform, `1033.mst`.
- `AGREETOLICENSE="YES"` this parameter must be included, or the installation can not complete.
- `ADDLOCAL="Server"` this parameter lists which components to install.

Procedure

1. Use the command line to conduct a silent installation.

- a) To invoke the silent installation from an elevated command prompt, click the **Start button** on your **Windows taskbar** and type `cmd` in **search programs and files** field. Right click the `cmd.exe` program and select **Run as administrator**.
- b) In the Windows command prompt, enter the following command:

Note: The command is presented on multiple lines here, but it must be typed out on one line.

```
msiexec /i "MQ_INSTALLATION_MEDIA\MSI\IBM MQ.msi"  
/l*v c:\wmqinslogs\install.log  
/q USEINI="MQ_INSTALLATION_MEDIA\Response.ini"  
TRANSFORMS="1033.mst"  
AGREETOLICENSE="yes"  
ADDLOCAL="Server"
```

Where `MQ_INSTALLATION_MEDIA` is the path to your IBM MQ installation media.

Note: The folder where you want the `install.log` to be created must exist before you run the command.

After you input the command, the command line will return the prompt.

- c) To view the installations progress, open the log file that you specified. If the installation completed successfully, you see the message `Product: IBM MQ (Installation1) -- Installation operation completed successfully.` two paragraphs up from the bottom of the log file.
- d) When the installation is complete, the service starts and the IBM MQ icon appears in the system tray.

You have installed IBM MQ, and you have started the IBM MQ service.

2. Set up environment variables for your installation by using the `setmqenv` command.

- a) Enter the following command in the command line:

```
"MQ_INSTALLATION_PATH/bin/setmqenv" -s
```

where `MQ_INSTALLATION_PATH` refers to the location where IBM MQ is installed. Ensure you enclose the path to **setmqenv** in the bin folder, in quotation marks, to prevent the prompt returning an error.

Note: If you used the default location, the path to your installation will be `C:\Program Files\IBM\MQ`.

- b) Check that the environment is set up correctly by entering the following command:

```
dspmqrver
```

If the command completes successfully, and the expected version number and installation name are returned, the environment is correctly set up. The message should include the line:

```
Version: n.n.n.n
```

where `n.n.n.n` is the version number and, if you did not specify a non-default installation name, the line:

```
InstName: Installation1
```

You have successfully installed IBM MQ using a silent installation.

Results

You have conducted an IBM MQ silent installation and confirmed that your environment is set up correctly.

What to do next

- You can run the [Prepare IBM MQ Wizard](#).
- Follow the instructions in [“Creating a queue manager called QM1”](#) on page 13.

If you encounter any issues during the installation, check the installation log, at the location that you specified in the **msiexec** command, in this scenario the location of the log file is: `c:\wmqinslogs\install.log`. Take any action that is specified in the log and rerun the installation again. You can also check the parameters that you passed with the command, making sure you are including all the required parameters.

Related tasks

[Installing the server using msiexec](#)

[Using transforms with msiexec](#)

[Installing IBM MQ - overview](#)

Creating a queue manager called QM1

Create a queue manager, called QM1 by using the command-line interface. Queue managers are the main components in an IBM MQ messaging network.

Before you begin

You must have IBM MQ installed. If you do not, see [“Installing using a silent installation”](#) on page 11 for information about how to do so.

About this task

In this example, all names are typed in uppercase and because IBM MQ names are case-sensitive, you must type all names in uppercase too.

Procedure

1. Open a command prompt as an administrator.
2. Create a queue manager with the name QM1 by typing the following command:

```
crtmqm QM1
```

When the system creates the queue manager, the following output is displayed:

```
C:\>crtmqm QM1
IBM MQ queue manager created.
Creating or replacing default objects for QM1.
Default objects statistics : 61 created. 0 replaced. 0 failed.
Completing setup.
Setup completed.
```

The queue manager is created, and is stopped. You must start the queue manager before you can administer it, and before you can read and write messages from its queues.

3. Start the queue manager by entering the following command:

```
strmqm QM1
```

When the queue manager successfully starts, the following output is displayed:

```
C:\>strmqm QM1
IBM MQ queue manager 'QM1' starting.
5 log records accessed on queue manager 'QM1' during the log replay phase.
Log replay for queue manager 'QM1' complete.
Transaction manager state recovered for queue manager 'QM1'.
IBM MQ queue manager 'QM1' started.
```

The queue manager is started.

What to do next

To create a queue, see [“Creating a queue called LQ1” on page 14](#).

Related tasks

[Creating and managing queue managers on Multiplatforms](#)

Creating a queue called LQ1

Create a queue by using the command-line interface. Queues are data structures that are used to store messages and are IBM MQ queue manager objects.

About this task

There are three ways to create IBM MQ objects:

- Command-line.
- IBM MQ Explorer.
- Using a programmable interface.

In this task you can create IBM MQ objects using the command-line.

The command-line interface has a scripting language called IBM MQ Script Commands (MQSC). The scripting tool, **runmqsc**, is used to run the script against a queue manager. To create and start a queue by using the command-line interface, complete the following steps.

Procedure

1. Start the scripting tool by typing the following command:

```
runmqsc QM1
```

When the scripting tool starts, the following output is displayed:

```
C:\>runmqsc QM1
5724-H72 (C) Copyright IBM Corp. 1994, 2024. ALL RIGHTS RESERVED.
Starting MQSC for queue manager QM1.
```

The tool is ready to accept MQSC commands.

2. Create a local queue called LQ1 by typing the following MQSC command:

```
define qlocal(LQ1)
```

When the queue is created, the following output is displayed:

```
define qlocal(LQ1)
2 : define qlocal(LQ1)
AMQ8006: IBM MQ queue created.
```

3. Stop the scripting tool by typing the following MQSC command:

```
end
```

When the scripting tool ends, the following output is displayed:

```
One MQSC command read.
No commands have a syntax error.
All valid MQSC commands were processed.

C:\>
```

What to do next

You are ready to put a message on to your queue. To put a message in a queue, see [“Putting a message to the queue LQ1” on page 15](#).

Putting a message to the queue LQ1

Put a message on to the queue LQ1 by using the command-line interface.

About this task

IBM MQ comes with a sample application called **amqspu**t. This application puts a message to a predefined queue.

To put a message to the queue by using the command-line interface, complete the following steps.

Procedure

1. Use the **amqspu**t sample application to put a message to queue LQ1, by typing the following command:

```
amqspout LQ1 QM1
```

When the sample application starts, the following output is displayed:

```
C:\>amqspout LQ1 QM1
Sample AMQSPUT0 start
target queue is LQ1
```

2. Type **Hello World** and press **Enter**. You placed a message that contains the text "Hello World" on the queue LQ1 managed by the queue manager called QM1.
3. To end **amqspout**, press **Enter**. The following output is displayed:

```
C:\>amqspout LQ1 QM1
Sample AMQSPUT0 start
target queue is LQ1
Hello World

Sample AMQSPUT0 end
```

What to do next

To get a message from the queue, see [“Getting a message from the queue LQ1” on page 16](#).

Getting a message from the queue LQ1

Get a message from the queue LQ1 by using the command-line interface.

About this task

IBM MQ comes with a sample application called **amqsget**. This application reads messages from a queue. To get a message from the queue by using the command-line interface, complete the following steps.

Procedure

Use the **amqsget** sample application to read a message on the queue LQ1, by typing the following command:

```
amqsget LQ1 QM1
```

When the sample application starts, the following output is displayed:

```
C:\>amqsget LQ1 QM1
Sample AMQSGET0 start
message <Hello World>
no more messages
Sample AMQSGET0 end
```

The **amqsget** application ends 30 seconds after reading the message.

What to do next

Follow the instructions in subsequent scenarios to explore further IBM MQ features.

To learn about writing queuing applications, connecting to and disconnecting from a queue manager, publish/subscribe, and opening and closing objects, see [Writing a procedural application for queuing](#).

Uninstalling IBM MQ

Stop, and then uninstall IBM MQ, including removing any queue managers and their objects. At the end of this task, you are ready to reinstall IBM MQ.

About this task

This task describes the steps for uninstalling IBM MQ on Windows by using the downloaded installation image.

The Getting started scenario takes you through options for installing IBM MQ by using the launchpad or command line. Although you can have more than one installation of IBM MQ, this scenario is based on a new installation on a single server. Therefore, if you want to repeat the scenario, or try out a different installation method, you must first uninstall the existing IBM MQ components, including any existing queue managers and their objects, so that you can start again with a fresh installation.

You might also need to uninstall so that you can carry out a fresh installation for some of the other scenarios in this section.

Procedure

1. Stop the IBM MQ service.

- a) Right-click on the **IBM MQ** icon in the system tray, then click **Stop IBM MQ** to stop the IBM MQ service.

A dialog with the following message is displayed:

Shutting down IBM MQ installation "Installation1" terminates all running queue managers and IBM MQ processes for that installation, except those under Microsoft Failover Cluster control.
Are you sure you want to continue?

- b) Click **Yes** and then wait for IBM MQ to stop.
- c) When IBM MQ stops, right-click the **IBM MQ** icon in the system tray, then click **Exit**

2. Begin the uninstalling process.

Download the compressed file that contains the installation image, then uncompress it into a temporary directory. Navigate to that directory, then double-click `setup.exe`.

The IBM MQ **Installation Launchpad** window is displayed.

3. Remove IBM MQ.

- a) Click **IBM MQ Installation**.
- b) Click **Launch IBM MQ Installer** and click **Next** until the IBM MQ **Program Maintenance pane** is displayed with a welcome message.
If this pane is not displayed, IBM MQ for Windows is not currently installed.
- c) Click **Maintain or upgrade an existing instance**. Select **Installation1** to remove it. Click **Next** and in the **Program Maintenance pane**, click **Remove**, then **Next**.

The Removing Server feature pane is shown.

- d) Select **Remove**: remove existing queue managers and their objects.

Click **Next**.

The Remove IBM MQ pane is displayed, with a summary of the installation to be removed.

- e) Click **Remove** to continue.

If a message appears stating that locked files are found, ensure that no IBM MQ programs are running; see [Uninstalling IBM MQ on Windows systems](#).

When IBM MQ is uninstalled, a message indicates completion.

- f) Click **Finish**.


You have successfully uninstalled IBM MQ.

Related tasks

[Uninstalling IBM MQ on Windows systems](#)

What to do next

What to do next on completion of the Getting started with IBM MQ scenario.

 For tutorials to help you with installing and upgrading, see [A collection of tutorials for installing and upgrading IBM MQ on AIX®, Linux®, and Windows](#). The tutorials cover:

- Preparing a host for IBM MQ.
- Downloading the IBM MQ code.
- Installing and uninstalling the IBM MQ code, and applying fix packs.
- Upgrading from one version of IBM MQ to another, and moving a queue manager from one host to another.

There are additional topics for you to view in the IBM MQ product documentation. You might want to look at the following sections:

- [Administering IBM MQ](#)

IBM MQ provides control commands that you can use. You use two of these commands in this scenario: **crtmqm** and **strmqm**. This section also provides a good overview about message queuing.

- [Administering IBM MQ using MQSC commands](#)

In this scenario, you use the `define qlocal('LQ1')` command to define a local queue called LQ1. This command is an MQSC command. IBM MQ system administrators use these commands to manage their queue managers. This section introduces the commands and shows you how to use them. The commands are described in detail, in alphabetical order, in the [MQSC commands](#) reference section.

- [Configuring a queue manager cluster](#)

This section describes how to organize, use, and manage queue managers in virtual groups known as clusters. Clustering ensures that each queue manager within a cluster knows about all the other queue managers in the same cluster. Clustering also makes the management of complex queue manager networks simpler.

Point-to-point scenario

Connect two IBM MQ queue managers in a point-to-point topology to enable distributed queuing.

About this task

Create two queue managers and the appropriate queues and channels to create a one-way, point-to-point messaging infrastructure. Create the queue managers on separate hosts to enable communication over a network. As an extension to the scenario, add Transport Layer Security to the channel to enable secure communication of data.

Planning the solution

Point-to-point messaging is the simplest form of messaging in IBM MQ. In point-to-point messaging, the sending application must know certain information about the receiving application before messages can be sent. The sending application will require a way to address the remote queue. Use point-to-point messaging to send a message to a remote queue manager with a sample application.

Overview: The delivered logical topology

The delivered logical topology after completing the scenario.

The point-to-point infrastructure allows one directional messaging between queue managers on different host machines. Queue manager one, on host one sends messages to queue manager two, on host two. After this scenario is complete, the delivered topology will look like [Figure 1](#).

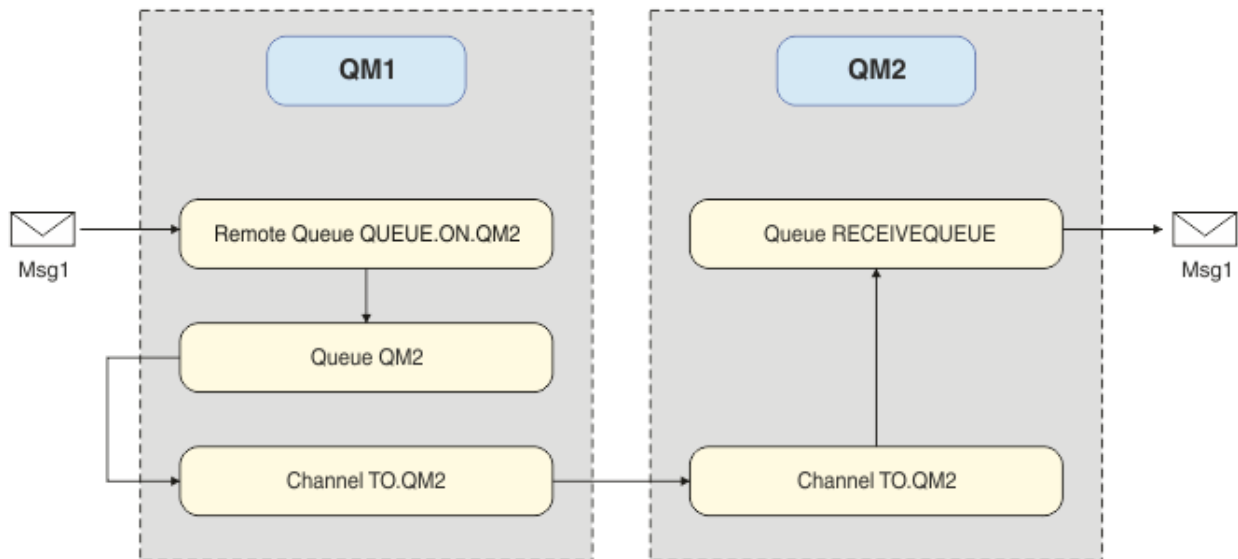


Figure 2. QM1 sends a message to QM2

Basic concepts and key terms

Descriptions of the basic concepts and key terms you must know to complete the point to point scenario.

Basic concepts

IBM MQ enables applications to read and write messages to a queue. The application that reads the message is independent of the application that writes the message. It is not a requirement to have the two applications running at the same time. If no application is available to read the message it is queued on the IBM MQ queue until an application reads it.

Key terms

Here is a list of key terms about message queuing.

Key terms about message queuing.

Term	Description
Queue managers	The queue manager is responsible for maintaining the queues it owns, and for storing all the messages it receives onto the appropriate queues.
Messages	A message is a string of bytes that is meaningful to the applications that use it. Messages are used to transfer information from one application program to another. The applications can be running on the same or on different computers.
Local queues	A local queue is a data structure used to store messages. The queue can be a normal queue or a transmission queue. A normal queue holds messages that are to be read by an application that is reading the message directly from the queue manager. A transmission queue holds messages that are in transit to another queue manager.
Remote queues	A remote queue is used to address a message to another queue manager.
Channels	Channels are used to send and receive messages between queue managers.
Listeners	Listeners are processes that accept network requests from other queue managers, or client applications, and start associated channels.

Implementing the solution

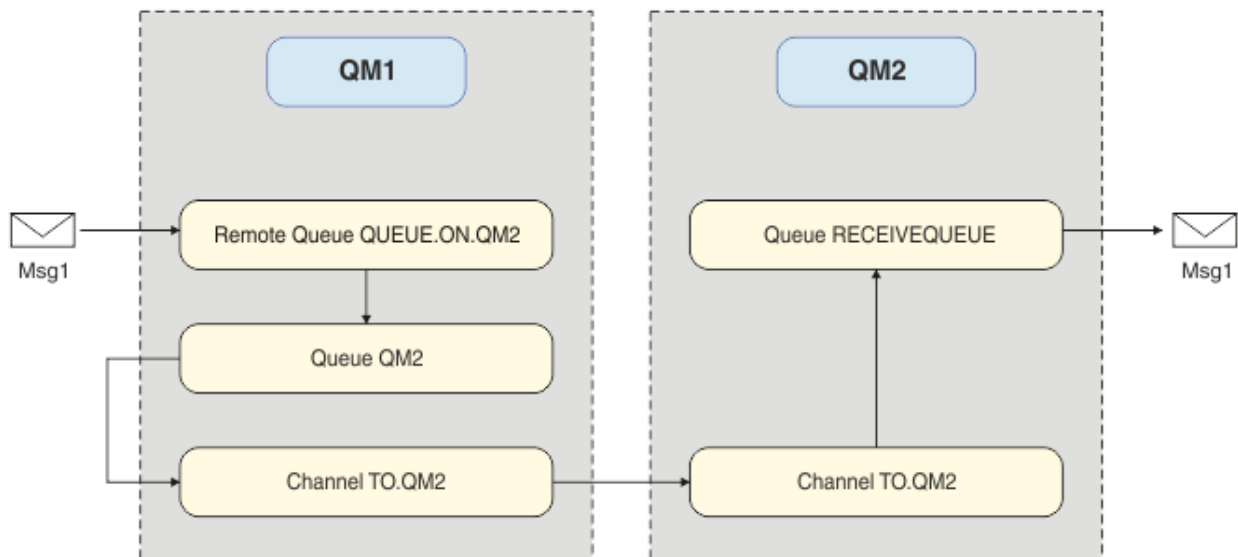
Implement the solution to the scenario. Create two IBM MQ queue managers on two separate hosts, the source queue manager to send messages, and the target queue manager to receive messages.

Before you begin

The starting point for this scenario is an existing, verified IBM MQ installation. For instructions to install IBM MQ, follow the steps in [Installing an IBM MQ server on Windows](#).

About this task

Create two queue managers by using the command-line interface, define the required listeners, queues, and channels. The delivered logical topology shows the functions added by implementing the solution.



Creating the queue manager

Create an IBM MQ queue manager to send messages to the target queue manager.

Before you begin

- You must have IBM MQ installed. For more information about installing IBM MQ, see [Installing and uninstalling](#).

About this task

Create the IBM MQ queue manager by using the command-line interface.

Procedure

1. Create a queue manager with the name QM1. On the command-line, type:

```
crtmqm QM1
```

The following messages are displayed to confirm that the queue manager is created:

```
IBM MQ queue manager created.  
Creating or replacing default objects for QM1.  
Default objects statistics : 61 created. 0 replaced. 0 failed.  
Completing setup.  
Setup completed.
```

2. Start the queue manager. On the command-line, type:

```
strmqm QM1
```

The following messages are displayed to confirm that the queue manager is started:

```
IBM MQ queue manager 'QM1' starting.  
5 log records accessed on queue manager 'QM1' during the log replay phase.  
Log replay for queue manager 'QM1' complete.  
Transaction manager state recovered for queue manager 'QM1'.  
IBM MQ queue manager 'QM1' started.
```

Results

The IBM MQ queue manager QM1 is created and started.

What to do next

To create the queues to use with QM1, follow the instructions in [“Creating the queues”](#) on page 21.

Creating the queues

Create IBM MQ queues that are managed by the IBM MQ queue manager.

Before you begin

You must have an IBM MQ queue manager that is set up as described in [“Creating the queue manager”](#) on page 21.

About this task

Start the **MQSC** interface to administer objects that are connected to the queue manager. Create a transmission queue, and a remote queue definition. Exit the **MQSC** interface.

Procedure

1. On the command-line, type:

```
runmqsc QM1
```

After a confirmation message, the tool is ready to accept commands.

2. Create a transmission queue called QM2. It is good practice to give the transmission queue the same name as the remote queue manager. In the MQSC interface, type:

```
DEFINE QLOCAL(QM2) DESC('Transmission queue to QM2') USAGE(XMITQ)
```

The transmission queue is created.

3. Create a remote queue definition called QUEUE.ON.QM2. The remote queue definition must refer to the name given to the local queue on the remote host. In the MQSC interface, type:

```
DEFINE QREMOTE(QUEUE.ON.QM2) DESC('Remote queue for QM2') XMITQ(QM2) RNAME(RECEIVEQUEUE)  
RQMNAME(QM2)
```

The remote queue definition is created.

4. Type end to exit the MQSC interface.

What to do next

To create the sender channel that is used to connect to the target queue manager, follow the instructions in [“Creating the sender channel” on page 22](#).

Creating the sender channel

Create the sender channel on the source queue manager, the channel is used to connect to the target queue manager.

Before you begin

To create a channel that uses TLS, follow the instructions in [“Creating the channels to use TLS” on page 28](#). This can be done afterward if you want to test the solution without TLS security.

About this task

Start the **MQSC** interface to administer objects that are connected to the queue manager and create the sender channel. This channel is used to connect to the target queue manager called QM2.

Procedure

1. On the command-line, type:

```
runmqsc QM1
```

After a confirmation message, the tool is ready to accept commands.

2. Create the sender channel, called TO.QM2. In the MQSC interface, type:

```
DEFINE CHANNEL(TO.QM2) CHLTYPE(SDR) CONNAME(' remoteHost ') TRPTYPE(TCP) XMITQ(QM2)
```

Note: The variable *remoteHost* is the host name or IP address of the target queue manager.

The sender channel is created.

What to do next

To create the distributed queue manager topology, follow the instructions in [“Creating the distributed queue manager topology”](#) on page 23.

Creating the distributed queue manager topology

Point-to-point messaging is the simplest form of messaging in IBM MQ. In point-to-point messaging, the sending application must know certain information about the receiving application before messages can be sent. The sending application will require a way to address the remote queue. Use point-to-point messaging to send a message to a second queue manager with a sample application.

Before you begin

You must have set up the source queue manager as described in [“Creating the queue manager”](#) on page 21.

About this task

Create the target queue manager on a remote host. Use the sample applications to verify communication between the source and target queue managers.

Creating the queue manager

Create an IBM MQ queue manager to receive messages from the remote queue manager.

Before you begin

You must have IBM MQ installed. For more information about installing IBM MQ, see [Installing an IBM MQ server on Windows](#).

About this task

Create the IBM MQ queue manager by using the command-line interface.

Procedure

1. Create a queue manager with the name QM2. On the command-line, type:

```
crtmqm QM2
```

The following messages are displayed:

```
IBM MQ queue manager created.  
Creating or replacing default objects for QM2.  
Default objects statistics : 61 created. 0 replaced. 0 failed.  
Completing setup.  
Setup completed.
```

2. Start the queue manager. On the command-line, type:

```
strmqm QM2
```

The following messages are displayed to confirm that the queue manager is started:

```
IBM MQ queue manager 'QM2' starting.  
5 log records accessed on queue manager 'QM2' during the log replay phase.  
Log replay for queue manager 'QM2' complete.  
Transaction manager state recovered for queue manager 'QM2'.  
IBM MQ queue manager 'QM2' started.
```

Results

The IBM MQ queue manager QM2 is created and started.

What to do next

To create the queue to use with QM2, follow the instructions in [“Creating the queue” on page 24](#).

Creating the queue

Create the local queue that is used to receive messages on the target queue manager, and the listener that accepts the inbound channel connection.

About this task

After you have started the **runmqsc** scripting tool, you can use MQSC commands to create a local queue and listener.

Procedure

1. Start the scripting tool by typing the following command:

```
runmqsc QM2
```

A message is displayed to confirm that the tool has started.

2. Create a local queue called RECEIVEQUEUE. The queue must have the same name as referred to in the remote queue definition on the source queue manager. In the MQSC interface, type:

```
DEFINE QLOCAL(RECEIVEQUEUE) DESCR('Receiving queue')
```

The local queue is created.

3. Create a listener called LISTENER1. In the MQSC interface, type:

```
DEFINE LISTENER(LISTENER1) TRPTYPE(TCP) PORT(1414) CONTROL(QMGR)
```

Note: Port 1414 is the default port for IBM MQ. If you chose a different port number, you must add it to the CONNAME of the sender channel on the sending queue manager.

4. Start the listener so it is ready to accept inbound connections. In the MQSC interface, type:

```
START LISTENER(LISTENER1)
```

Note: Since the listener was created with the option CONTROL(QMGR), next time the queue manager is started, the listener will also be automatically started.

5. Type end to exit the **MQSC** interface.

What to do next

To create the receiver channel to create the connection between the source and target queue managers, follow the instructions in [“Creating the receiver channel” on page 24](#).

Creating the receiver channel

Create the receiver channel for the target queue manager to enable communication between the source and target queue managers.

Before you begin

To create a channel that uses TLS, follow the instructions in [“Creating the channels to use TLS” on page 28](#). This can be done afterward if you want to test the solution without TLS security.

About this task

Use the **MQSC** interface to create a receiver channel that is managed by QM2.

Procedure

1. On the command-line, type:

```
runmqsc QM2
```

After a confirmation message, the tool is ready to accept commands.

2. Create a receiver channel called TO.QM2. The channel must have the same name as the sender channel on the source queue manager. In the MQSC interface, type:

```
DEFINE CHANNEL(TO.QM2) CHLTYPE(RCVR) TRPTYPE(TCP)
```

The receiver channel is created.

What to do next

To start the sender channel on the source queue manager, that in turn initiates the receiver channel on the target queue manager, follow the instructions in [“Starting the sender channel” on page 25](#).

Starting the sender channel

Start the sender channel on the source queue manager, the receiver channel on the target queue manager is also started. Messages can be sent from the source queue manager to the target queue manager.

About this task

Start the **MQSC** interface to administer objects that are connected to the queue manager. Start the sender channel to connect to the target queue manager, enabling communication. The receiver channel starts automatically when the source channel is started.

Procedure

1. On the command-line, type:

```
runmqsc QM1
```

After a confirmation message, the tool is ready to accept commands.

2. Start the sender channel on the source queue manager. In the MQSC interface, type:

```
START CHANNEL(TO.QM2)
```

The sender channel starts, the receiver channel on the target queue manager is also started.

3. Check that the channel is running. In the MQSC interface, type:

```
DISPLAY CHSTATUS(TO.QM2)
```

If the channel is running, you will see that it reports STATUS(RUNNING). If it reports any other value in STATUS then check the [error log](#).

What to do next

To verify that the source queue manager can send messages to the target queue manager, follow the instructions in [“Verifying the solution” on page 26](#).

Verifying the solution

Verify that the source queue manager can put a message onto the remote queue. Verify that the target queue manager can get the message from the queue.

About this task

Use the sample applications, **amqspout** and **amqsget** to verify the solution.

Procedure

1. Send a message to the target queue manager, QM2 from the source queue manager.

- a) In the command-line interface, type:

```
amqspout QUEUE.ON.QM2 QM1
```

You must use the name of the remote queue definition to send the message to the target queue manager.

The following message is displayed:

```
Sample AMQSPUT0 start  
target queue is QUEUE.ON.QM2
```

- b) Type `Hello world.`, press Enter twice.
2. Get the message on the target queue manager.

- a) In the command-line interface, type:

```
amqsget RECEIVEQUEUE QM2
```

The following message is displayed:

```
Sample AMQSGET0 start  
message <Hello world.>  
no more messages  
Sample AMQSGET0 end
```

Results

The target queue manager received the message from the source queue manager, verifying that point to point communication is achieved.

What to do next

If you want to add security to the solution, follow the instructions in [“Securing the point-to-point topology”](#) on page 26.

Securing the point-to-point topology

Secure the point-to-point topology so that messages can be transmitted in a production environment.

About this task

Secure the source and target queue manager objects so that the correct level of access is granted. Define which user groups have access to the queues and queue managers. Secure the network connection by using digitally signed certificates to connect using Transport Layer Security (TLS).

Securing the source queue manager objects

Set the authorization values for the objects on the source queue manager.

About this task

Use the **setmqaut** command to grant authorities to the user group running the application.

Procedure

1. To grant the specified user group with *connect* authorization to the queue manager, on the command-line interface, type:

```
setmqaut -m QM1 -t qmgr -g userGroup +connect
```

2. To grant the specified user group with *put* authorization on the remote queue definition, on the command-line interface, type:

```
setmqaut -m QM1 -t q -n "QUEUE.ON.QM2" -g userGroup +put
```

Securing the target queue manager objects

Set the authorization values for the objects on the target queue manager.

About this task

Use the **setmqaut** command to grant authorities to the user group running the application.

Procedure

1. To grant the specified user group with *connect* authorization to the queue manager, on the command-line interface, type:

```
setmqaut -m QM2 -t qmgr -g userGroup +connect
```

2. To grant the specified user group with *get* authorization on the remote queue definition, in the command-line interface, type:

```
setmqaut -m QM2 -t q -n "RECEIVEQUEUE" -g userGroup +get
```

Securing the network

Secure the network connections between the source and remote queue managers.

About this task

Use signed certificates to verify the authenticity of the source and remote queue managers. Transfer messages using a TLS network to encrypt messages.

Preparing the queue managers to use TLS

The IBM MQ queue manager's key repository is used to store the queue manager's personal certificate and the public Certificate Authority (CA) certificate. The personal certificate request from the IBM MQ queue manager must be signed by a CA, the public certificate is used by the other entities to authenticate the IBM MQ queue manager.

Before you begin

You must have the public Certificate Authority certificate in a file.

About this task

Create the IBM MQ queue manager's key repository, import the certificate authority's signer certificate and create the queue manager's personal certificate request.

Procedure

1. Create a CMS key repository file for the queue manager called `key.kdb`. Navigate to the `Qmgrs\QM1\ssl` directory, and on the command line, type:

```
runmqakm -keydb -create -db key.kdb -pw passw0rd -type cms -stash
```

Note: For this simple example we have used a password of `passw0rd`. You may wish to choose a different password and change each of the following commands to use your own password instead.

2. Add the CA certificate, which you have in a file, to the key repository, on the command line, type:

```
runmqakm -cert -add -file CA-certificate-file -db key.kdb -pw passw0rd -label TrustedCA
```

3. Request a personal certificate that will be written to a request file called `QM1req.req`.

On the command line, enter:

```
runmqakm -certreq -create -db key.kdb -pw passw0rd -label ibmwebsphermqqm1  
-dn CN="QM1" -size 1024 -file QM1req.req -sig_alg SHA1WithRSA
```

The default certificate label name is shown in this example. You can set your own name if you prefer. For details, see [Digital certificate labels](#).

4. Send the certificate request file to your CA, they will issue a digitally signed certificate. Put the received, signed certificate file in a suitable location to be received into the queue manager's key repository.
5. Receive the signed personal certificate into the queue manager's key repository.

```
runmqakm -cert -receive -file Signed-certificate-file -db key.kdb -pw passw0rd -format ascii
```

6. Complete these steps for each queue manager, changing the queue manager name accordingly.

What to do next

To enable secure communication over the sender and receiver channels, follow the instructions in [“Creating the channels to use TLS”](#) on page 28.

Creating the channels to use TLS

Create a new channel that uses TLS to create a connection.

Before you begin

To communicate over a channel that uses TLS, first you must have the required certificates for each end of the connection. To create the required certificates, follow the instructions in [“Preparing the queue managers to use TLS”](#) on page 27.

About this task

Use the MQSC interface to define channels with TLS attributes set. This task can be done even if you defined your channels without TLS in a prior step through the use of the `REPLACE` keyword.

Procedure

1. On the command-line, type:

```
runmqsc QM1
```

2. Create the sender channel on QM1, called `T0.QM2`, in the MQSC interface, type:

```

DEFINE CHANNEL(TO.QM2) CHLTYPE(SDR) TRPTYPE(TCP)
CONNAME('remoteHost') XMITQ(QM2)
SSLCIPH(TLS_RSA_WITH_AES_128_CBC_SHA256)
DESCR('Sender channel using TLS from QM1 to QM2')
REPLACE

```

Note: The variable *remoteHost* is the host name or IP address of the target queue manager.

You can specify a CERTLABL attribute for the channel. If you do, it must match the value on the **-label** parameter of the **runmqakm** command that you previously ran in step “3” on page 28 of “Preparing the queue managers to use TLS” on page 27. For more information on certificate labels, see [Digital certificate labels, understanding the requirements](#).

3. Type end to exit the MQSC interface.
4. On the command-line, type:

```
runmqsc QM2
```

5. Create a receiver channel on QM2, called TO.QM2, in the MQSC interface, type:

```

DEFINE CHANNEL(TO.QM2) CHLTYPE(RCVR) TRPTYPE(TCP)
SSLCIPH(TLS_RSA_WITH_AES_128_CBC_SHA256) SSLCAUTH(REQUIRED)
DESCR('Receiver channel using TLS from QM1 to QM2')
REPLACE

```

6. Type end to exit the MQSC interface.

What to do next

To verify that the source queue manager can send messages to the target queue manager using TLS, follow the instructions in “[Verifying the solution](#)” on page 26.

Streaming queues

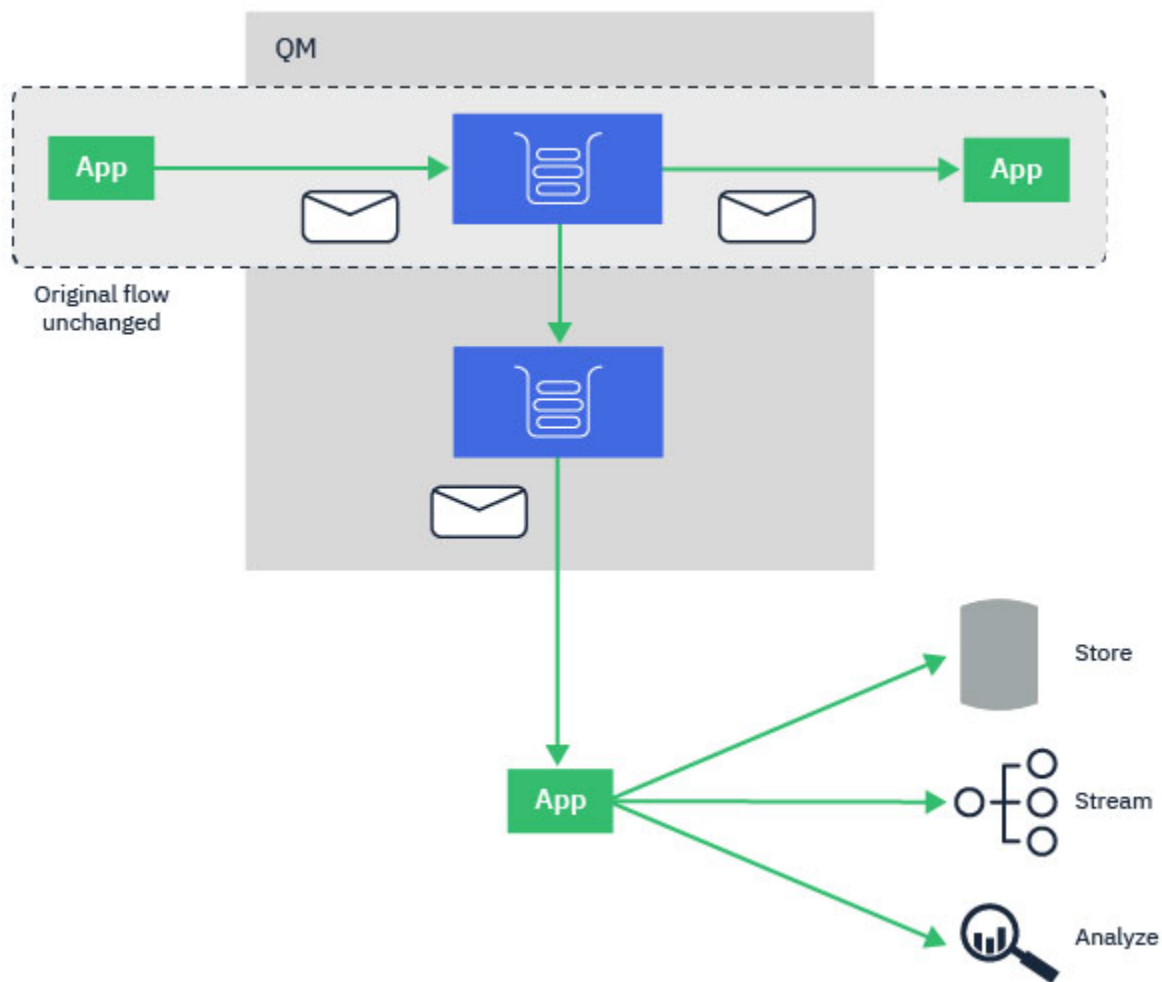
The streaming queues feature of IBM MQ allows you to configure a queue to put a near-identical copy of every message to a second queue.

Streaming queues can be useful in certain scenarios, where you need to create a copy of your messages. For example:

- Streaming messages to Apache Kafka using the Kafka Connect source connector for IBM MQ. For more information, see [kafka_connect_mq_source](#).
- Performing analysis on the data going through the system.
- Storing messages for recovery at a later time.
- Capturing a set of messages to use in development and test systems.
- Consuming IBM MQ event messages from the system event queues, and sending additional copies to other queues or topics.

In all of these scenarios, you can configure streaming queues to ensure that the original messages remain unaffected by the streaming process. This ensures that core business applications do not observe any impact from the streaming.

The following illustration depicts this:



Related concepts

[Streaming queues security](#)

[Streaming queues and AMS](#)

Streaming queues configuration

The streaming queues feature of IBM MQ is configured by the administrator on individual queues, and the messages are streamed by the queue manager, not by the application itself.

This means that in almost all cases the application putting messages to the original queue is completely unaware that streaming is taking place. Similarly, the application consuming messages from the original queue is unaware that message streaming has taken place.

Note: The version of the IBM MQ client library does not need upgrading to make use of streaming queues, and the original messages are completely unchanged by the streaming process.

You can configure streaming queues in one of two modes:

Best effort

In this mode, the queue manager considers it more important that delivery of the original message is not affected by delivery of the streamed message.

If the original message can be delivered, but the streamed message cannot, the original message is still delivered to its queue. This mode is best suited to those applications, where it is important for the original business application to remain unaffected by the streaming process.

Must duplicate

In this mode, the queue manager ensures that both the original message and the streamed message are successfully delivered to their queues.

If, for some reason, the streamed message cannot be delivered to its queue, for example, because the second queue is full, then the original message is not delivered to its queue either. The putting application receives an error reason code and must try to put the message again.

See [How you configure streaming queues](#) for information on the additional attributes added to local and model queues enabling message streaming.

Streamed messages

In most cases, the copy of the message delivered to the second queue is a duplicate of the original message. This includes all of the message descriptor fields, including the message ID and correlation ID. The streamed messages are intended to be very close copies of the original messages, so that they are easier to find and, if necessary, replay them back into another IBM MQ system.

There are some message descriptor fields that are not retained on the streamed message. The following changes are made to the streamed message before it is placed on the second queue:

- The expiry of the streamed message is set to MQEI_UNLIMITED, regardless of the expiry of the original message. If **CAPEXPRI** has been set on the secondary queue its value is used to set the expiry time of the streamed message.
- If any of the following report options are set on the original message, they are not enabled on the streamed message. This is to ensure that no unexpected report messages are delivered to applications that are not designed to receive them:
 - Activity reports
 - Expiration reports
 - Exception reports
 - Confirmation on arrival (COA)
 - Confirmation on delivery (COD)

Due to the near-identical nature of the streamed messages, most of the attributes of the secondary queue have no affect on the message descriptor fields of the streamed message. For example, the **DEFPSIST** and **DEFPRTY** attributes of the secondary queue have no affect on the streamed message.

The following exceptions apply to the streamed message:

- **CAPEXPRI** attribute

If the secondary queue has been configured with a **CAPEXPRI** attribute, this expiry cap is applied to the expiry of the streamed message.

- **DEFBIND** for cluster queues

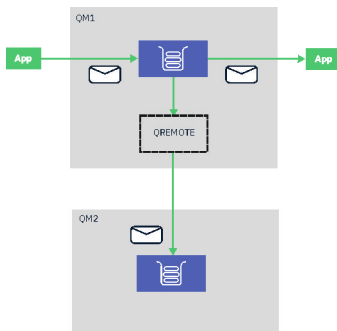
If the secondary queue is a cluster queue, the streamed message is put using the bind option set in the **DEFBIND** attribute of the secondary queue.

Streaming to remote and alias queues

It is possible to stream messages to remote queues and alias queues. For example, Q1 might be configured with `STREAMQ(MY.REMOTE.Q)` where MY.REMOTE.Q is a remote queue definition.

Streaming to remote queues

By streaming messages from a local queue to a remote queue, the duplicate messages can be sent to a queue on another queue manager in your IBM MQ network as shown in the following illustration:



Streaming to alias queues

By streaming messages to an alias queue, it is possible to send the duplicate messages to the target of the alias queue. Since the target of an alias queue can also be a topic, it is therefore possible to send the duplicate messages to a publish/subscribe topic. Any subscribers to the alias topic will receive a copy of the duplicate message. In this way, you can create multiple copies of the original message. However, the existing rules for publish/subscribe message are applied to the duplicated message. This means that the messages that are sent to subscribers will not be identical to the original message, including:

- Having a new message ID.
- Having a generated correlation ID, depending on the configuration of the subscription.
- The UserIdentifier field being set to the user the queue manager is running as, not the user who put the message.
- The PutApplName showing the name of the queue manager, not the name of the putting application.

Notes:

1. It is not possible to configure the **STREAMQ** attribute on remote queues or alias queues themselves. You can only stream messages to them, not from them.
2. If messages are being streamed to a queue alias, the target of the queue alias cannot have its **STREAMQ** attribute set.

Streaming queue restrictions

Certain configurations are not supported when using streaming queues in IBM MQ, and these are documented here.

The following list specifies the configurations that are not supported:

- Defining a chain of queues streaming to each other, for example, Q1->Q2, Q2->Q3, Q3->Q4
- Defining a loop of streaming queues, for example, Q1->Q2, Q2->Q1
- Defining a subscription with a provided destination, where that destination has a STREAMQ defined
- Defining STREAMQ on a queue configured with USAGE(XMITQ)

Note: STREAMQ can be a remote queue, but you cannot configure the STREAMQ attribute on a remote queue definition.

- Modifying the STREAMQ attribute of a dynamic queue
- Setting STREAMQ to any value that begins SYSTEM.*, except for SYSTEM.DEFAULT.LOCAL.QUEUE
- Defining STREAMQ on any queue named SYSTEM.*, with the following exceptions:
 - SYSTEM.DEFAULT.LOCAL.QUEUE
 - SYSTEM.ADMIN.ACCOUNTING.QUEUE
 - SYSTEM.ADMIN.ACTIVITY.QUEUE
 - SYSTEM.ADMIN.CHANNEL.EVENT
 - SYSTEM.ADMIN.COMMAND.EVENT

- SYSTEM.ADMIN.CONFIG.EVENT
- SYSTEM.ADMIN.LOGGER.EVENT
- SYSTEM.ADMIN.PERFM.EVENT
- SYSTEM.ADMIN.PUBSUB.EVENT
- SYSTEM.ADMIN.QMGR.EVENT
- SYSTEM.ADMIN.STATISTICS.QUEUE
- SYSTEM.DEFAULT.MODEL.QUEUE
- SYSTEM.JMS.TEMPQ.MODEL
- Setting STREAMQ to the name of a model queue

Stream queues and transactions

The streaming queues feature allows a message put to one queue, to be duplicated to a second queue. In most cases the two messages are put to their respective queues under a unit of work.

If the original message was put using MQPMO_SYNCPOINT, the duplicate message is put to the stream queue under the same unit of work that was started for the original put.

If the original was put with MQPMO_NO_SYNCPOINT, a unit of work will be started even though the original put did not request one. This is done for two reasons:

1. It ensures the duplicate message is not delivered if the original message could not be. The streaming queues feature only delivers messages to stream queues if the original message was also delivered.
2. There can be a performance improvement by doing both puts inside a unit of work

The only time the messages are not delivered inside a unit of work is when the original MQPUT is non-persistent with MQPMO_NO_SYNCPOINT, and the **STRMQOS** attribute of the queue is set to BESTEF (best effort).

Notes:

1. The additional put to the stream queue does not count towards the MAXUMSGS limit.
2. In the case of a queue configured with STRMQOS(BESTEF), failure to deliver the duplicate message does not cause the unit of work to be rolled back.

Streaming to and from cluster queues

It is possible to stream messages from a local queue to a cluster queue and to stream messages from cluster queue instances to a local queue.

Streaming to a cluster queue

This can be useful if you have a local queue where original messages are delivered, and would like to stream a copy of every message to one or more instances of a cluster queue. This could be to workload balance the processing of the duplicate messages, or simply to have duplicate messages streamed to another queue elsewhere in the cluster.

When streaming messages to a cluster queue, messages are distributed using the cluster workload balancing algorithm. A cluster queue instance is chosen based on the DEFBIND attribute of the cluster queue.

For example, if the cluster queue is configured with DEFBIND(OPEN), an instance of the cluster queue is chosen when the original queue is opened. All duplicate messages go to the same cluster queue instance, until the original queue is reopened by the application.

If the cluster queue is configured with DEFBIND(NOTFIXED), an instance of the cluster queue will be chosen for every MQPUT operation.

Note: You should configure all cluster queue instances with the same value for the DEFBIND attribute.

Streaming from a cluster queue

This can be useful if you already send messages to several instances of a cluster queue, and would like a copy of each message to be delivered to a streaming queue, on the same queue manager, as the cluster queue instance.

When the original message is delivered to one of the cluster queue instances, a duplicate message is delivered to the stream queue by the cluster-receiver channel.

Multi

V 9.4.0

Using streaming queues to store a history of messages

You can use streaming queues to retain a history of messages for a limited period of time, and you achieve this by configuring the CAEXPRY attribute on the queue that messages are streamed to.

Introduction

When messages are streamed from one queue to another, any expiry value set on the message is reset to a value of MQEI_UNLIMITED for the duplicate copy. By default, this results in a steady build up of messages on the queue you are streaming to, if there is no application consuming them.

In this scenario you want to keep a copy of the messages for a limited period of time so that you are able to access them. For example, if the original message has been accidentally removed by a consuming application.

It is not feasible to keep a copy of every message indefinitely, and to prevent the queue you are streaming to from becoming full there are two options:

- Run an application to remove the messages every so often
- Configure the messages with an expiry, causing them to be removed by IBM MQ after a period of time.

The second option can be much more convenient as it does not require you to run and maintain an application, just to prevent the queue from filling up.

Configuring CAEXPRY

The [Enforcing lower expiration times](#) topic describes how to configure CAEXPRY on a queue. For this scenario, you must set the CAEXPRY attribute on the queue to which you are streaming messages.

Note: You do not need to change the value of the CAEXPRY attribute on the queue that messages are being streamed from.

Choose a suitable expiry time for the duplicate messages, taking into account the following considerations:

1. How long you might need access to the messages for
2. What the MAXDEPTH attribute of the queue needs to be, based on the rate of messages being put to the original queue
3. How much storage you need in order to store the duplicate messages.

This might require consideration of the queue manager file system size, and the MAXFSIZE attribute of the queue.

Accessing the duplicate messages if they are needed

If a problem should occur that requires you to access and recover some or all the duplicate messages, use the **dmpmqmsg** command, to help access those messages.

dmpmqmsg has options for:

- Reading the messages from the queue and writing a copy to file to access later
- Reading messages from a file and writing them back onto a queue for applications to consume

It is possible to edit the message headers once **dmpmqmsg** has written them to a file. For example, you could reset the expiry of the messages back to MQEI_UNLIMITED, before **dmqmqmsg** puts them back onto a queue for processing, by changing the EXP value of every message in the file to -1.

Performance considerations

There is a small cost to streaming duplicate messages to another queue and expiring them when they are no longer required. However, the cost is much smaller than manually putting a copy to a second queue and having an application destructively remove them after a period of time. The [Streaming Queues Performance Report](#) gives more information about the performance of this scenario.

Related concepts

[Streaming queues security](#)

[Streaming queues and AMS](#)

Publish/subscribe scenarios

Two sets of scenarios that demonstrate use of publish/subscribe clusters and publish/subscribe hierarchies.

The available publish/subscribe scenarios are described in the following subtopics:

Scenario: Creating a publish/subscribe cluster

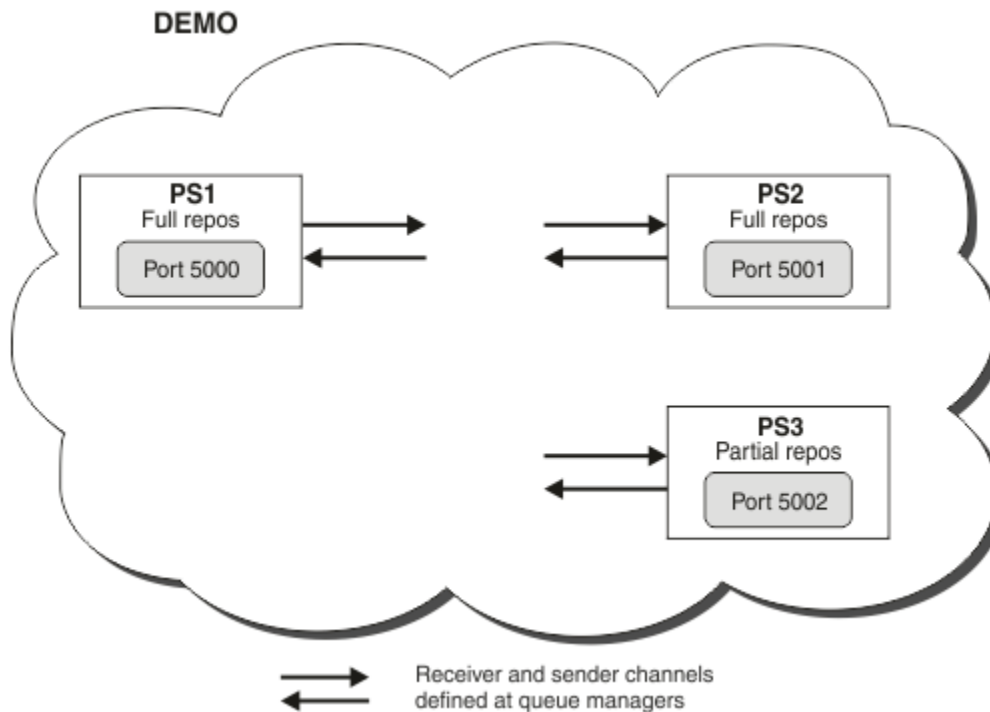
In this scenario you create a simple three queue manager cluster and configure it to allow subscriptions created on one queue manager to receive messages published by an application connected to another queue manager.

Before you begin

The starting point for this scenario is an existing IBM MQ installation. For instructions to install IBM MQ , follow the steps in [Installing an IBM MQ server on Windows](#).

About this task

By completing the steps in this scenario, you first create the following cluster:



This cluster consists of three queue managers, two of which are defined as full repository queue managers.

You then define a cluster topic on queue manager PS3. By creating the cluster topic, you have made the cluster into a publish/subscribe cluster. To test the publish/subscribe cluster, you subscribe to the topic on any queue manager, then publish a message to the topic from another queue manager and check that your subscription receives the message.

Related tasks

[Designing publish/subscribe clusters](#)

[Configuring a queue manager cluster](#)

Creating and starting the queue managers

Create and start three queue managers, called PS1, PS2 and PS3.

Procedure

1. Create and start queue manager PS1.

- a) Create the queue manager.

In the command line, enter the following command:

```
crtmqm PS1
```

- b) Start the queue manager.

In the command line, enter the following command:

```
strmqm PS1
```

2. Repeat step 1 to create and start queue manager PS2.

3. Repeat step 1 to create and start queue manager PS3.

What to do next

You are now ready to [configure the first queue manager](#).

Configuring the first queue manager

Use the MQSC interface to define a listener and a receiver channel for PS1, to set the queue manager as a full repository for the cluster, and to define a sender channel from PS1 to PS2 so the two full repositories can exchange information.

Before you begin

This task assumes that you have completed the steps in [“Creating and starting the queue managers”](#) on page 36.

Procedure

1. Define and start a listener for PS1.

- a) Launch the MQSC interface.

In the command line, enter the following command:

```
runmqsc PS1
```

- b) Define a listener.

Enter the following MQSC command:

```
DEFINE LISTENER(PS1_LS) TRPTYPE(TCP) CONTROL(QMGR) PORT(5000)
```

- c) Start the listener.

Enter the following MQSC command:

```
START LISTENER(PS1_LS)
```

2. Set the queue manager as a full repository for the cluster.

Enter the following MQSC command:

```
ALTER QMGR REPOS(DEMO)
```

3. Define a receiver channel for PS1, to allow other queue managers in the cluster to communicate with it.

Enter the following MQSC command:

```
DEFINE CHANNEL(DEMO.PS1) CHLTYPE(CLUSRCVR) TRPTYPE(TCP) CONNAME('$HOSTNAME(5000)')  
CLUSTER(DEMO)  
DESCR('TCP Cluster-receiver channel for queue manager PS1')
```

4. Define a sender channel from PS1 to PS2, to allow the two full repositories to exchange information.

Enter the following MQSC command:

```
DEFINE CHANNEL(DEMO.PS2) CHLTYPE(CLUSSDR) TRPTYPE(TCP) CONNAME('$HOSTNAME(5001)')  
CLUSTER(DEMO)  
DESCR('TCP Cluster-sender channel from PS1 to queue manager PS2')
```

What to do next

You are now ready to [configure the second queue manager](#).

Configuring the second queue manager

Use the MQSC interface to define a listener and a receiver channel for PS2, to set the queue manager as a full repository for the cluster, and to define a sender channel from PS2 to PS1 so the two full repositories can exchange information.

Before you begin

This task assumes that you have completed the steps in [“Configuring the first queue manager”](#) on page 37.

Procedure

1. Define and start a listener for PS2.

- a) Launch the MQSC interface.

In the command line, enter the following command:

```
runmqsc PS2
```

- b) Define a listener.

Enter the following MQSC command:

```
DEFINE LISTENER(PS2_LS) TRPTYPE(TCP) CONTROL(QMGR) PORT(5001)
```

- c) Start the listener.

Enter the following MQSC command:

```
START LISTENER(PS2_LS)
```

2. Set the queue manager as a full repository for the cluster.

Enter the following MQSC command:

```
ALTER QMGR REPOS(DEMO)
```

3. Define a receiver channel for PS2, to allow other queue managers in the cluster to communicate with it.

Enter the following MQSC command:

```
DEFINE CHANNEL(DEMO.PS2) CHLTYPE(CLUSRCVR) TRPTYPE(TCP) CONNAME('$HOSTNAME(5001)')  
CLUSTER(DEMO)  
DESCR('TCP Cluster-receiver channel for queue manager PS2')
```

4. Define a sender channel from PS2 to PS1, to allow the two full repositories to exchange information.

Enter the following MQSC command:

```
DEFINE CHANNEL(DEMO.PS1) CHLTYPE(CLUSSDR) TRPTYPE(TCP) CONNAME('$HOSTNAME(5000)')  
CLUSTER(DEMO)  
DESCR('TCP Cluster-sender channel from PS2 to PS1')
```

What to do next

You are now ready to [configure the third queue manager](#).

Configuring the third queue manager

Use the MQSC interface to define a listener and a receiver channel for PS3. Join PS3 into the cluster by defining a sender channel from PS3 to one of the full repository queue managers.

Before you begin

This task assumes that you have completed the steps in [“Configuring the second queue manager”](#) on page 38.

Procedure

1. Define and start a listener for PS3.

- a) Launch the MQSC interface.

In the command line, enter the following command:

```
runmqsc PS3
```

- b) Define a listener.

Enter the following MQSC command:

```
DEFINE LISTENER(PS3_LS) TRPTYPE(TCP) CONTROL(QMGR) PORT(5002)
```

- c) Start the listener.

Enter the following MQSC command:

```
START LISTENER(PS3_LS)
```

2. Define a receiver channel for PS3, to allow other queue managers in the cluster to communicate with it.

Enter the following MQSC command:

```
DEFINE CHANNEL(DEMO.PS3) CHLTYPE(CLUSRCVR) TRPTYPE(TCP) CONNAME('$HOSTNAME(5002)')  
CLUSTER(DEMO)  
DESCR('TCP Cluster-receiver channel for queue manager PS3')
```

3. Define a sender channel from PS3 to one of the full repository queue managers (for example, PS1). This joins PS3 into the cluster.

Enter the following MQSC command:

```
DEFINE CHANNEL(DEMO.PS1) CHLTYPE(CLUSSDR) TRPTYPE(TCP) CONNAME('$HOSTNAME(5000)')  
CLUSTER(DEMO)  
DESCR('TCP Cluster-sender channel from PS3 to PS1')
```

4. Validate that PS3 has successfully joined the cluster.

Enter the following MQSC command:

```
DISPLAY CLUSQMGR(*) QMTYPE
```

This command returns three entries, one each for QM1, QM2 and QM3. QM1 and QM2 should have a **QMTYPE** of REPOS, and QM3 should have a **QMTYPE** of NORMAL.

What to do next

You are now ready to [define a cluster topic](#).

Defining cluster topics

Publishing and subscribing applications can publish to any topic string, with no need for an administered topic object to be defined. However, if the publishing applications are connected to a cluster queue manager that is different to the queue managers where subscriptions are created, an administered topic object must be defined and added to the cluster. To make a topic a cluster topic, you specify the name of the cluster in its definition.

Before you begin

This task assumes that you have completed the steps in [“Configuring the third queue manager” on page 39](#).

About this task

The administered topic object identifies the point in the topic tree that is clustered through its topic string. Publishing and subscribing applications can use any topic string at or below that point, and their messages are automatically transmitted between queue managers.

When you define a cluster topic, you also choose its routing model. For more information about publication routing in clusters, see [Designing publish/subscribe clusters](#).

For this scenario we use the default routing of *DIRECT*. This means that messages are sent direct from a publishing queue manager to the subscribing queue managers.

Procedure

1. Define the cluster topic SCORES on PS3.

To make the topic a cluster topic, specify the name of the cluster, and set the cluster routing (**CLROUTE**) that you want to use for publications and subscriptions for this topic.

- a) Launch the MQSC interface.

In the command line, enter the following command:

```
runmqsc PS3
```

- b) Define the cluster topic SCORES.

Enter the following MQSC command:

```
DEFINE TOPIC(SCORES) TOPICSTR('/Sport/Scores') CLUSTER(DEMO) CLROUTE(DIRECT)
```

- c) Enter end to exit the MQSC interface for PS3.

2. Verify the topic definition on PS1.

- a) Launch the MQSC interface for PS1.

In the command line, enter the following command:

```
runmqsc PS1
```

- b) Display the cluster state for cluster topic SCORES.

Enter the following MQSC command:

```
DISPLAY TCLUSTER(SCORES) CLSTATE
```

The **CLSTATE** for cluster topic SCORES is shown as ACTIVE.

What to do next

For a more detailed exploration of this task, see [Configuring a publish/subscribe cluster](#).

You are now ready to verify the solution. See [“Testing the publish/subscribe cluster” on page 41](#).

Testing the publish/subscribe cluster

Test the publish/subscribe cluster by publishing and subscribing to a topic string from different queue managers in the cluster.

Before you begin

This task assumes that you have completed the steps in [“Defining cluster topics” on page 40](#).

About this task

Using the command line, and the amqspub and amqssub sample applications that are included with IBM MQ, you can publish a topic from one queue manager and subscribe to the topic with the other queue managers. When a message is published to the topic, it is received by the subscribing queue managers.

Procedure

1. In the command line, enter the following command:

```
amqspub /Sport/Scores/Football PS1
```

2. Concurrently, in separate command lines, enter the following commands:

```
amqssub /Sport/Scores/Football PS2
```

```
amqssub /Sport/Scores/Football PS3
```

3. In the first command line, enter a message.

The message is displayed in both the subscribing command lines.

Note: The amqssub application will time out if a publication is not received for ten seconds.

Results

The publish/subscribe cluster set up is complete.

What to do next

Try defining different topic objects for different branches of the topic tree, and with different routing models.

Publish/subscribe hierarchy scenarios

Three scenarios that demonstrate use of publish/subscribe hierarchies. Each of the three scenarios sets up the same simple publish/subscribe topology. In each scenario, the queue managers rely on a different method for connecting to their neighboring queue managers in the hierarchy.

The available publish/subscribe hierarchy scenarios are described in the following subtopics:

Related concepts

[Publish/subscribe hierarchies](#)

Publish/subscribe hierarchy scenario 1: Using point-to-point channels with queue manager name alias

This is the first in a set of three scenarios that set up a publish/subscribe hierarchy in different ways to establish the connection between queue managers. This scenario sets up a publish/subscribe hierarchy that uses point-to-point channels with queue manager name alias.

About this task

This set of scenarios all use a parent queue manager called QM1, and two child queue managers called QM2, and QM3.

Scenario 1 is split into smaller sections to make the process easier to follow.

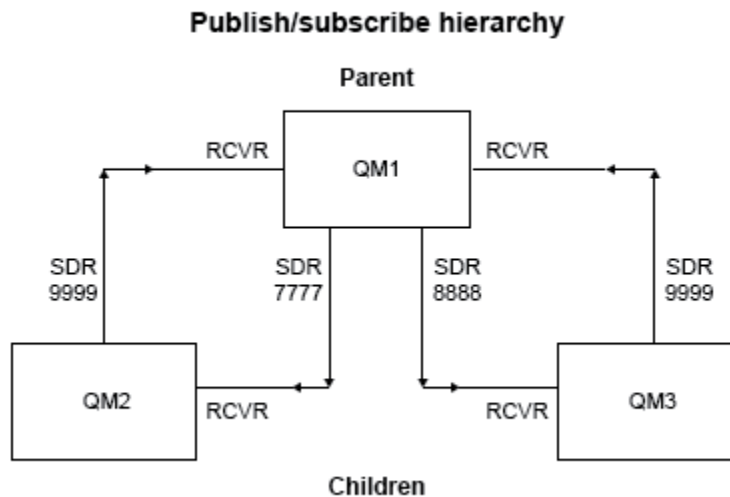


Figure 3. Topology diagram showing the relationship between queue managers in a typical publisher/subscribe hierarchy.

Procedure

1. Create the queue managers.

- a) Create and start three queue managers called QM1, QM2, and QM3 using the following commands:

```
crtmqm -u SYSTEM.DEAD.LETTER.QUEUE QM1
strmqm QM1

crtmqm -u SYSTEM.DEAD.LETTER.QUEUE QM2
strmqm QM2

crtmqm -u SYSTEM.DEAD.LETTER.QUEUE QM3
strmqm QM3
```

- b) Enable the queue manager publish/subscribe mode by using the following command on all three queue managers:

```
ALTER QMGR PSMODE(ENABLED)
```

2. Establish point-to-point channel connections between queue managers using a queue manager alias with the same name as the parent queue manager.

- a) Define a transmission queue and queue manager alias on QM2 to QM1. Define a sender channel to QM1 and a receiver channel for the sender channel created on QM1 for QM2:

```

DEFINE QLOCAL(QM1.XMITQ) USAGE(XMITQ)

DEFINE QREMOTE (QM1) RNAME('') RQMNAME(QM1) XMITQ(QM1.XMITQ)

DEFINE CHANNEL('QM2.TO.QM1') CHLTYPE(SDR) CONNAME('localhost(9999)') XMITQ(QM1.XMITQ)
TRPTYPE(TCP)

DEFINE CHANNEL('QM1.TO.QM2') CHLTYPE(RCVR) TRPTYPE(TCP)

```

- b) Define a transmission queue and queue manager alias on QM3 to QM1. Define sender channel to QM1 and a receiver channel for the sender channel created on QM1 for QM3:

```

DEFINE QLOCAL(QM1.XMITQ) USAGE(XMITQ)

DEFINE QREMOTE (QM1) RNAME('') RQMNAME(QM1) XMITQ(QM1.XMITQ)

DEFINE CHANNEL('QM3.TO.QM1') CHLTYPE(SDR) CONNAME('localhost(9999)') XMITQ(QM1.XMITQ)
TRPTYPE(TCP)

DEFINE CHANNEL('QM1.TO.QM3') CHLTYPE(RCVR) TRPTYPE(TCP)

```

- c) Define a transmission queue and queue manager alias on QM1 to QM2 and QM3. Define sender channel to QM2 and QM3, and a receiver channel for the sender channels created on QM2 and QM3 for QM1:

```

DEFINE QLOCAL(QM2.XMITQ) USAGE(XMITQ)

DEFINE QREMOTE (QM2) RNAME('') RQMNAME(QM2) XMITQ(QM2.XMITQ)

DEFINE CHANNEL('QM1.TO.QM2') CHLTYPE(SDR) CONNAME('localhost(7777)') XMITQ(QM2.XMITQ)
TRPTYPE(TCP)

DEFINE CHANNEL('QM2.TO.QM1') CHLTYPE(RCVR) TRPTYPE(TCP)

DEFINE QLOCAL(QM3.XMITQ) USAGE(XMITQ)

DEFINE QREMOTE (QM3) RNAME('') RQMNAME(QM3) XMITQ(QM3.XMITQ)

DEFINE CHANNEL('QM1.TO.QM3') CHLTYPE(SDR) CONNAME('localhost(8888)') XMITQ(QM3.XMITQ)
TRPTYPE(TCP)

DEFINE CHANNEL('QM3.TO.QM1') CHLTYPE(RCVR) TRPTYPE(TCP)

```

- d) Start the appropriate listeners on the queue managers:

```

runmqclsr -m QM1 -t TCP -p 9999 &
runmqclsr -m QM2 -t TCP -p 7777 &
runmqclsr -m QM3 -t TCP -p 8888 &

```

- e) Start the following channels:

- i) On QM1:

```

START CHANNEL('QM1.TO.QM2')
START CHANNEL('QM1.TO.QM3')

```

- ii) On QM2:

```

START CHANNEL('QM2.TO.QM1')

```

- iii) On QM3:

```

START CHANNEL('QM3.TO.QM1')

```

- f) Check that all the channels have started:

```

DISPLAY CHSTATUS('QM1.TO.QM2')
DISPLAY CHSTATUS('QM1.TO.QM3')
DISPLAY CHSTATUS('QM2.TO.QM1')
DISPLAY CHSTATUS('QM3.TO.QM1')

```

g)

3. Connect the queue managers and define a topic.

Connect the child queue managers QM2 and QM3 to the parent queue manager QM1.

a) On QM2 and QM3, set the parent queue manager to QM1:

```
ALTER QMGR PARENT (QM1)
```

b) Run the following command on all queue managers to check that the child queue managers are connected to the parent queue manager:

```
DISPLAY PUBSUB TYPE(ALL)
```

Command output is displayed. For example, here is output for QM1, with the key details highlighted:

```

DISPLAY PUBSUB ALL
1 : DISPLAY PUBSUB ALL
AMQ8723: Display pub/sub status details.
QMNAME(QM1) TYPE(LOCAL)
STATUS(ACTIVE) SUBCOUNT(6)
TPCOUNT(9)
AMQ8723: Display pub/sub status details.
QMNAME(QM2) TYPE(CHILD)
STATUS(ACTIVE) SUBCOUNT(NONE)
TPCOUNT(NONE)
AMQ8723: Display pub/sub status details.
QMNAME(QM3) TYPE(CHILD)
STATUS(ACTIVE) SUBCOUNT(NONE)
TPCOUNT(NONE)

```

4. Use the amqspub.exe and amqssub.exe applications to publish and subscribe the topic.

a) Run this command in the first command window:

```
amqspub Sport/Soccer QM2
```

b) Run this command in the second command window:

```
amqssub Sport/Soccer QM1
```

c) Run this command in the third command window:

```
amqssub Sport/Soccer QM3
```

Results

The amqssub.exe applications in the second and third command windows receive the messages published in the first command window.

Publish/subscribe hierarchy scenario 2: Using point-to-point channels with same name for transmission queue and remote queue manager

This is the second in a set of three scenarios that set up a publish/subscribe hierarchy in different ways to establish the connection between queue managers. This scenario sets up a publish/subscribe hierarchy

that uses point-to-point channels with the transmission queue name the same as the remote queue manager.

About this task

This set of scenarios all use a parent queue manager called QM1, and two child queue managers called QM2, and QM3.

This scenario reuses steps 1, 3, and 4 from [“Publish/subscribe hierarchy scenario 1: Using point-to-point channels with queue manager name alias”](#) on page 42.

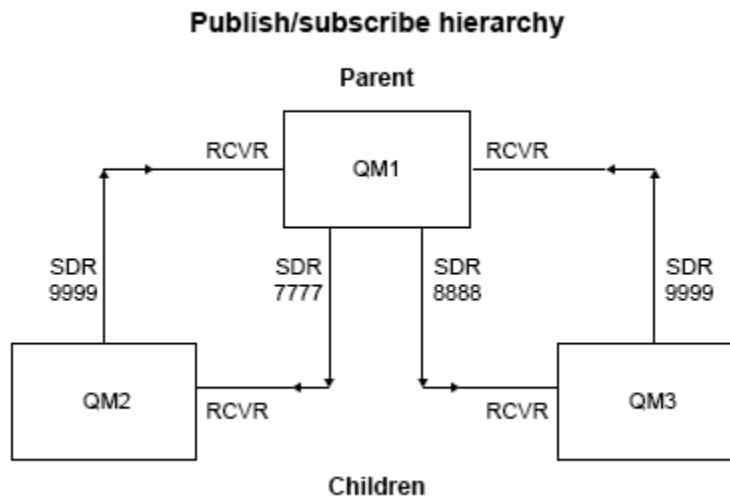


Figure 4. Topology diagram showing the relationship between queue managers in a typical publisher/subscribe hierarchy.

Procedure

1. Create the queue managers.
 - a) Create and start three queue managers called QM1, QM2, and QM3 using the following commands:

```
crtmqm -u SYSTEM.DEAD.LETTER.QUEUE QM1
strmqm QM1

crtmqm -u SYSTEM.DEAD.LETTER.QUEUE QM2
strmqm QM2

crtmqm -u SYSTEM.DEAD.LETTER.QUEUE QM3
strmqm QM3
```

- b) Enable the queue manager publish/subscribe mode by using the following command on all three queue managers:

```
ALTER QMGR PSMODE(ENABLED)
```

2. Establish point-to-point channel connections between a queue manager using a transmission queue with the same name as the parent queue manager.
 - a) Define a transmission queue on QM2 to QM1. Define a sender channel to QM1 and a receiver channel for the sender channel for QM2 created on QM1:

```
DEFINE QLOCAL(QM1) USAGE(XMITQ)

DEFINE CHANNEL('QM2.TO.QM1') CHLTYPE(SDR) CONNAME('localhost(9999)') XMITQ(QM1)
```

```
TRPTYPE(TCP)

DEFINE CHANNEL('QM1.TO.QM2') CHLTYPE(RCVR) TRPTYPE(TCP)
```

- b) Define a transmission queue on QM3 to QM1. Define sender channel to QM1 and a receiver channel for the sender channel created on QM1 for QM3:

```
DEFINE QLOCAL(QM1) USAGE(XMITQ)

DEFINE CHANNEL('QM3.TO.QM1') CHLTYPE(SDR) CONNAME('localhost(9999)') XMITQ(QM1)
TRPTYPE(TCP)

DEFINE CHANNEL('QM1.TO.QM3') CHLTYPE(RCVR) TRPTYPE(TCP)
```

- c) Define transmission queues on QM1 to QM2 and QM3. Define sender channels to QM2 and QM3, and a receiver channel for the sender channels created on QM2 and QM3 for QM1:

```
DEFINE QLOCAL(QM2) USAGE(XMITQ)

DEFINE CHANNEL('QM1.TO.QM2') CHLTYPE(SDR) CONNAME('localhost(7777)') XMITQ(QM2)
TRPTYPE(TCP)

DEFINE CHANNEL('QM2.TO.QM1') CHLTYPE(RCVR) TRPTYPE(TCP)

DEFINE QLOCAL(QM3) USAGE(XMITQ)

DEFINE CHANNEL('QM1.TO.QM3') CHLTYPE(SDR) CONNAME('localhost(8888)') XMITQ(QM3)
TRPTYPE(TCP)

DEFINE CHANNEL('QM3.TO.QM1') CHLTYPE(RCVR) TRPTYPE(TCP)
```

- d) Start the appropriate listeners on the queue managers:

```
runmqclsr -m QM1 -t TCP -p 9999 &
runmqclsr -m QM2 -t TCP -p 7777 &
runmqclsr -m QM3 -t TCP -p 8888 &
```

- e) Start the following channels:

- i) On QM1:

```
START CHANNEL('QM1.TO.QM2')
START CHANNEL('QM1.TO.QM3')
```

- ii) On QM2:

```
START CHANNEL('QM2.TO.QM1')
```

- iii) On QM3:

```
START CHANNEL('QM3.TO.QM1')
```

- f) Check that all the channels have started:

```
DISPLAY CHSTATUS('QM1.TO.QM2')
DISPLAY CHSTATUS('QM1.TO.QM3')
DISPLAY CHSTATUS('QM2.TO.QM1')
DISPLAY CHSTATUS('QM3.TO.QM1')
```

3. Connect the queue managers and define a topic.

Connect the child queue managers QM2 and QM3 to the parent queue manager QM1.

- a) On QM2 and QM3, set the parent queue manager to QM1:

```
ALTER QMGR PARENT (QM1)
```

- b) Run the following command on all queue managers to check that the child queue managers are connected to the parent queue manager:

```
DISPLAY PUBSUB TYPE(ALL)
```

Command output is displayed. For example, here is output for QM1, with the key details highlighted:

```
DISPLAY PUBSUB ALL
1 : DISPLAY PUBSUB ALL
AMQ8723: Display pub/sub status details.
QMNAME(QM1) TYPE(LOCAL)
STATUS(ACTIVE) SUBCOUNT(6)
TPCOUNT(9)
AMQ8723: Display pub/sub status details.
QMNAME(QM2) TYPE(CHILD)
STATUS(ACTIVE) SUBCOUNT(NONE)
TPCOUNT(NONE)
AMQ8723: Display pub/sub status details.
QMNAME(QM3) TYPE(CHILD)
STATUS(ACTIVE) SUBCOUNT(NONE)
TPCOUNT(NONE)
```

4. Use the amqspub.exe and amqssub.exe applications to publish and subscribe the topic.

- a) Run this command in the first command window:

```
amqspub Sport/Soccer QM2
```

- b) Run this command in the second command window:

```
amqssub Sport/Soccer QM1
```

- c) Run this command in the third command window:

```
amqssub Sport/Soccer QM3
```

Results

The amqssub.exe applications in the second and third command windows receive the messages published in the first command window.

Related tasks

[“Publish/subscribe hierarchy scenario 1: Using point-to-point channels with queue manager name alias” on page 42](#)

This is the first in a set of three scenarios that set up a publish/subscribe hierarchy in different ways to establish the connection between queue managers. This scenario sets up a publish/subscribe hierarchy that uses point-to-point channels with queue manager name alias.

[“Publish/subscribe hierarchy scenario 3: Using a cluster channel to add a queue manager” on page 48](#)

This is the third in a set of three scenarios that set up a publish/subscribe hierarchy in different ways to establish the connection between queue managers. This scenario uses a cluster channel to add a queue manager to a hierarchy.

[Connecting a queue manager to a publish/subscribe hierarchy](#)

Publish/subscribe hierarchy scenario 3: Using a cluster channel to add a queue manager

This is the third in a set of three scenarios that set up a publish/subscribe hierarchy in different ways to establish the connection between queue managers. This scenario uses a cluster channel to add a queue manager to a hierarchy.

About this task

This set of scenarios all use a parent queue manager called QM1, and two child queue managers called QM2, and QM3.

Note: This scenario is only using the cluster configuration to connect queue managers together, not to propagate publish/subscribe traffic through clustering topics. When defining child/parent hierarchy relationships between queue managers in the same cluster, propagation of publications between queue managers will occur based on the publication and subscription scope settings of the topics in the topic tree. It is important not to use the cluster name setting of a topic to add the topics into the cluster. If using the cluster name, the topology becomes a publish/subscribe cluster and does not require the child/parent hierarchy relationships defined. See [“Scenario: Creating a publish/subscribe cluster”](#) on page 35 and [Planning your distributed publish/subscribe network](#).

This scenario reuses steps 1, 3, and 4 from [“Publish/subscribe hierarchy scenario 1: Using point-to-point channels with queue manager name aliases”](#) on page 42.

This scenario creates a cluster called DEMO where QM1 and QM2 are full repositories, and QM3 is a partial repository. Queue manager QM1 is the parent of queue managers QM2 and QM3.

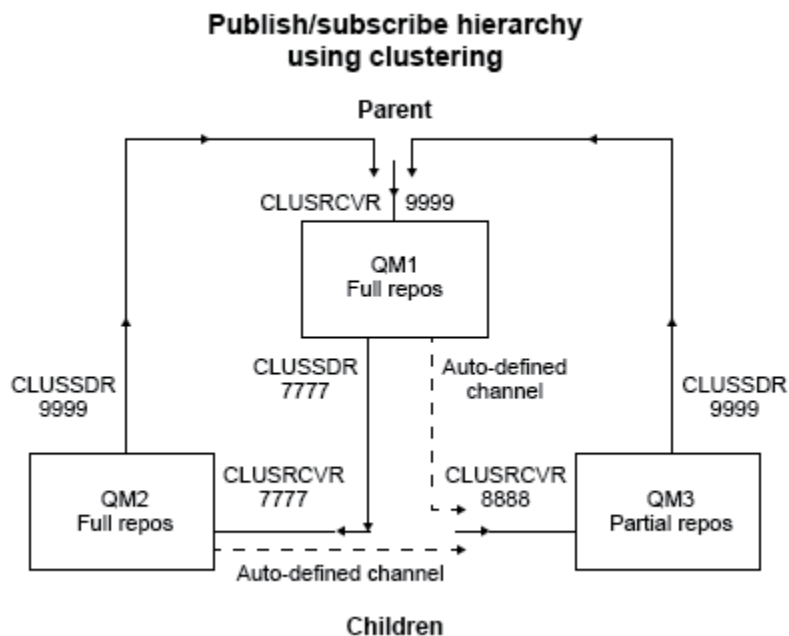


Figure 5. Topology diagram showing the relationship between queue managers that are using a cluster channel.

Procedure

1. Create the queue managers.
 - a) Create and start three queue managers called QM1, QM2, and QM3 using the following commands:

```
crtmqm -u SYSTEM.DEAD.LETTER.QUEUE QM1
startmq QM1
```



```

crtmqm -u SYSTEM.DEAD.LETTER.QUEUE QM2
stimqm QM2

crtmqm -u SYSTEM.DEAD.LETTER.QUEUE QM3
stimqm QM3

```

- b) Enable the queue manager publish/subscribe mode by using the following command on all three queue managers:

```
ALTER QMGR PSMODE(ENABLED)
```

2. Establish the point-to-point channel connections between queue managers a cluster.

- a) On QM1 and QM2, set the **REPOS** parameter to the name of the cluster DEMO:

```
ALTER QMGR REPOS(DEMO)
```

- b) Start the appropriate listeners on the queue managers:

```

runmqclsr -m QM1 -t TCP -p 9999 &
runmqclsr -m QM2 -t TCP -p 7777 &
runmqclsr -m QM3 -t TCP -p 8888 &

```

- c) Define the cluster receiver channel on each queue manager:

- i) On QM1:

```
DEFINE CHANNEL(TO.QM1) CHLTYPE(CLUSRCVR) TRPTYPE(TCP) CONNAME('localhost(9999)')
CLUSTER(DEMO)
```

- ii) On QM2:

```
DEFINE CHANNEL(TO.QM2) CHLTYPE(CLUSRCVR) TRPTYPE(TCP) CONNAME('localhost(7777)')
CLUSTER(DEMO)
```

- iii) On QM3:

```
DEFINE CHANNEL(TO.QM3) CHLTYPE(CLUSRCVR) TRPTYPE(TCP) CONNAME('localhost(8888)')
CLUSTER(DEMO)
```

- d) Define a cluster sender channel to a full repository on each queue manager in the cluster:

- i) On QM1:

```
DEFINE CHANNEL(TO.QM2) CHLTYPE(CLUSSDR) TRPTYPE(TCP) CONNAME('localhost(7777)')
CLUSTER(DEMO)
```

- ii) On QM2:

```
DEFINE CHANNEL(TO.QM1) CHLTYPE(CLUSSDR) TRPTYPE(TCP) CONNAME('localhost(9999)')
CLUSTER(DEMO)
```

- iii) QM3 can have a cluster sender channel to either full repository on QM1 or QM2. This example defines the channel to QM1:

```
DEFINE CHANNEL(TO.QM1) CHLTYPE(CLUSSDR) TRPTYPE(TCP) CONNAME('localhost(9999)')
CLUSTER(DEMO)
```

3. Connect the queue managers and define a topic.

Connect the child queue managers QM2 and QM3 to the parent queue manager QM1.

- a) On QM2 and QM3, set the parent queue manager to QM1:

```
ALTER QMGR PARENT (QM1)
```

- b) Run the following command on all queue managers to check that the child queue managers are connected to the parent queue manager:

```
DISPLAY PUBSUB TYPE(ALL)
```

Command output is displayed. For example, here is output for QM1, with the key details highlighted:

```
DISPLAY PUBSUB ALL
1 : DISPLAY PUBSUB ALL
AMQ8723: Display pub/sub status details.
QMNAME(QM1)          TYPE(LOCAL)
STATUS(ACTIVE)       SUBCOUNT(6)
TPCOUNT(9)
AMQ8723: Display pub/sub status details.
QMNAME(QM2) TYPE(CHILD)
STATUS(ACTIVE) SUBCOUNT(NONE)
TPCOUNT(NONE)
AMQ8723: Display pub/sub status details.
QMNAME(QM3) TYPE(CHILD)
STATUS(ACTIVE) SUBCOUNT(NONE)
TPCOUNT(NONE)
```

4. Use the `amqspub.exe` and `amqssub.exe` applications to publish and subscribe the topic.

- a) Run this command in the first command window:

```
amqspub Sport/Soccer QM2
```

- b) Run this command in the second command window:

```
amqssub Sport/Soccer QM1
```

- c) Run this command in the third command window:

```
amqssub Sport/Soccer QM3
```

Results

The `amqssub.exe` applications in the second and third command windows receive the messages published in the first command window.

Related tasks

[“Publish/subscribe hierarchy scenario 1: Using point-to-point channels with queue manager name alias” on page 42](#)

This is the first in a set of three scenarios that set up a publish/subscribe hierarchy in different ways to establish the connection between queue managers. This scenario sets up a publish/subscribe hierarchy that uses point-to-point channels with queue manager name alias.

[“Publish/subscribe hierarchy scenario 2: Using point-to-point channels with same name for transmission queue and remote queue manager” on page 44](#)

This is the second in a set of three scenarios that set up a publish/subscribe hierarchy in different ways to establish the connection between queue managers. This scenario sets up a publish/subscribe hierarchy that uses point-to-point channels with the transmission queue name the same as the remote queue manager.

[Connecting a queue manager to a publish/subscribe hierarchy](#)

Transactional support scenarios

Using transactional support you can enable your applications to work reliably with databases.

This section introduces transactional support. The work required to enable your applications to use IBM MQ with a database product spans the areas of application programming and system administration. Use the information here together with [Committing and backing out units of work](#).

We start by introducing the units of work that form transactions, then describe the ways in which you enable IBM MQ to coordinate transactions with databases.

Related concepts

[“Introducing units of work” on page 51](#)

This topic introduces and defines the general concepts of unit of work, commit, backout and sync point. It also contains two scenarios that illustrate global units of work.

Introducing units of work

This topic introduces and defines the general concepts of unit of work, commit, backout and sync point. It also contains two scenarios that illustrate global units of work.

When a program puts messages on queues within a unit of work, those messages are made visible to other programs only when the program *commits* the unit of work. To commit a unit of work, all updates must be successful to preserve data integrity.

If the program detects an error and decides not to make the put operation permanent, it can *back out* the unit of work. When a program performs a backout, IBM MQ restores the queues by removing the messages that were put on the queues by that unit of work.

Similarly, when a program gets messages from one or more queues within a unit of work, those messages remain on the queues until the program commits the unit of work, but the messages are not available to be retrieved by other programs. The messages are permanently deleted from the queues when the program commits the unit of work. If the program backs out the unit of work, IBM MQ restores the queues by making the messages available to be retrieved by other programs.

The decision to commit or back out the changes is taken, in the simplest case, at the end of a task. However, it can be more useful for an application to synchronize data changes at other logical points within a task. These logical points are called sync points (or synchronization points) and the period of processing a set of updates between two sync points is called a *unit of work*. Several MQGET calls and MQPUT calls can be part of a single unit of work.

With IBM MQ, we need to distinguish between *local* and *global* units of work:

Local units of work

Are those in which the only actions are puts to, and gets from, IBM MQ queues, and the coordination of each unit of work is provided within the queue manager using a *single-phase commit* process.

Use local units of work when the only resources to be updated are the queues that are managed by a single IBM MQ queue manager. Updates are committed by using the MQCMIT verb or backed out using MQBACK.

There are no system administration tasks, other than log management, which is involved in using local units of work. In your applications, where you use the MQPUT and MQGET calls with MQCMIT and MQBACK, try using the MQPMO_SYNCPOINT and MQGMO_SYNCPOINT options. (For information about log management, see [Managing log files](#).)

Global units of work

Are those in which other resources, such as tables in a relational database, are also updated. When more than one *resource manager* is involved, there is a need for *transaction manager* software that uses a *two-phase commit* process to coordinate the global unit of work.

Use global units of work when you also need to include updates to relational database manager software, such as Db2®, Oracle, Sybase, and Informix®.

There are several possible scenarios for using global units of work. Documented here are two scenarios:

1. In the first, the queue manager itself acts as the transaction manager. In this scenario, MQI verbs control the global units of work; they are started in applications using the MQBEGIN verb and then committed using MQCMIT or backed out using MQBACK.
2. In the second, the transaction manager role is performed by other software, such as TXSeries®, Encina, or Tuxedo. In this scenario, an API provided by the transaction manager software is used to control the unit of work (for example, EXEC CICS® SYNCPOINT for TXSeries).

The following sections describe all the steps necessary to use global units of work, organized by the two scenarios:

- [“Scenario 1: Queue manager performs the coordination” on page 52](#)
- [“Scenario 2: Other software provides the coordination” on page 76](#)

Multi

Scenario 1: Queue manager performs the coordination

In scenario 1, the queue manager acts as the transaction manager. In this scenario, MQI verbs control the global units of work; they are started in applications using the MQBEGIN verb and then committed using MQCMIT or backed out using MQBACK.

Multi

Isolation level

In IBM MQ, a message on a queue might be visible before a database update, depending on the transaction isolation design implemented within the database.

When an IBM MQ queue manager is working as an XA transaction manager, to coordinate updates to XA resource managers, the following commit protocol is followed:

1. Prepare all XA resource managers.
2. Commit the IBM MQ queue manager resource manager.
3. Commit other resource managers.

Between step 2 and 3, an application might see a message that is committed to the queue but the corresponding row in the database does not reflect this message.

This is not a problem if the database is configured such that the application's database API calls wait for pending updates to be completed.

You can resolve this by configuring the database differently. The type of configuration needed is referred to as the "isolation level". For more information on isolation levels, refer to the database documentation. You can, alternatively, configure the queue manager to commit the resource managers in the following reverse order:

1. Prepare all XA resource managers.
2. Commit other resource managers.
3. Commit the IBM MQ queue manager resource manager.

When you change the protocol the IBM MQ queue manager is committed last, so applications that read messages from the queues see a message only after the corresponding database update has been completed.

To configure the queue manager to use this changed protocol, set the **AMQ_REVERSE_COMMIT_ORDER** environment variable.

Set this environment variable in the environment from which the **strmqm** is run to start the queue manager. For example, run the following in the shell just before starting the queue manager:

```
export AMQ_REVERSE_COMMIT_ORDER=1
```

Note: Setting this environment variable might cause an extra log entry per transaction, so this will have a small impact on the performance of each transaction.

Multi Database coordination

When the queue manager coordinates global units of work itself, it becomes possible to integrate database updates within the units of work. That is, a mixed MQI and SQL application can be written, and the MQCMIT and MQBACK verbs can be used to commit or roll back the changes to the queues and databases together.

The queue manager achieves this using the two-phase commit protocol described in *X/Open Distributed Transaction Processing: The XA Specification*. When a unit of work is to be committed, the queue manager first asks each participating database manager whether it is prepared to commit its updates. Only if all the participants, including the queue manager itself, are prepared to commit, are all the queue and database updates committed. If any participant cannot prepare its updates, the unit of work is backed out instead.

In general, a global unit of work is implemented in an application by the following method (in pseudocode):

```
MQBEGIN
MQGET (include the flag MQGMO_SYNCPOINT in the message options)
MQPUT (include the flag MQPMO_SYNCPOINT in the message options)
SQL INSERT
MQCMIT
```

The purpose of MQBEGIN is to denote the beginning of a global unit of work. The purpose of MQCMIT is to denote the end of the global unit of work, and to complete it with all participating resource managers, using the two-phase commit protocol.

When the unit of work (also known as a *transaction*) is completed successfully using MQCMIT, all actions taken within that unit of work are made permanent or irreversible. If, for any reason, the unit of work fails, all actions are instead backed out. It is not possible for one action in a unit of work to be made permanent while another is backed out. This is the principle of a unit of work: either all actions within the unit of work are made permanent or none of them are.

Note:

1. The application programmer can force a unit of work to be backed out by calling MQBACK. The unit of work is also backed out by the queue manager if the application or database *fails* before MQCMIT is called.
2. If an application calls MQDISC without calling MQCMIT, the queue manager behaves as if MQCMIT had been called, and commits the unit of work.

In between MQBEGIN and MQCMIT, the queue manager does not make any calls to the database to update its resources. That is, the only way a database's tables are changed is by your code (for example, the SQL INSERT in the pseudocode).

Full recovery support is provided if the queue manager loses contact with any of the database managers during the commit protocol. If a database manager becomes unavailable while it is in doubt, that is, it has successfully prepared to commit, but has yet to receive a commit or backout decision, the queue manager remembers the outcome of the unit of work until that outcome has been successfully delivered to the database. Similarly, if the queue manager terminates with incomplete commit operations outstanding, these are remembered over queue manager restart. If an application terminates unexpectedly, the integrity of the unit of work is not compromised, but the outcome depends on where in the process the application terminated, as described in [Table 2 on page 54](#).

What happens when the database or application program fails is summarized in the following tables:

Table 1. What happens when a database server fails	
Failure occurrence	Outcome
Before the application call to MQCMIT.	The unit of work is backed out.

Table 1. What happens when a database server fails (continued)

Failure occurrence	Outcome
During the application call to MQCMIT, before all databases have indicated that they have successfully prepared.	The unit of work is backed out with a reason code of MQRC_BACKED_OUT.
During the application call to MQCMIT, after all databases have indicated that they have successfully prepared, but before all have indicated that they have successfully committed.	The unit of work is held in recoverable state by the queue manager, with a reason code of MQRC_OUTCOME_PENDING.
During the application call to MQCMIT, after all databases have indicated that they have successfully committed.	The unit of work is committed with a reason code of MQRC_NONE.
After the application call to MQCMIT.	The unit of work is committed with a reason code of MQRC_NONE.

Table 2. What happens when an application program fails

Failure occurrence	Outcome
Before the application call to MQCMIT.	The unit of work is backed out.
During the application call to MQCMIT, before the queue manager has received the application's MQCMIT request.	The unit of work is backed out.
During the application call to MQCMIT, after the queue manager has received the application's MQCMIT request.	The queue manager tries to commit using two-phase commit (subject to the database products successfully executing and committing their parts of the unit of work).

In the case where the reason code on return from MQCMIT is MQRC_OUTCOME_PENDING, the unit of work is remembered by the queue manager until it has been able to reestablish contact with the database server, and tell it to commit its part of the unit of work. Refer to “Considerations when contact is lost with the XA resource manager” on page 70 for information on how and when recovery is done.

The queue manager communicates with database managers using the XA interface as described in *X/Open Distributed Transaction Processing: The XA Specification*. Examples of these function calls are xa_open, xa_start, xa_end, xa_prepare, and xa_commit. We use the terms *transaction manager* and *resource manager* in the same sense as they are used in the XA specification.

Multi Restrictions

There are restrictions to the database coordination support.

The following restrictions apply:

- The ability to coordinate database updates within IBM MQ units of work is **not** supported in an MQI client application. The use of MQBEGIN in a client application fails. A program that calls MQBEGIN must run as a *server* application on the same machine as the queue manager.

Note: A *server* application is a program that has been linked with the necessary IBM MQ server libraries; a *client* application is a program that has been linked with the necessary IBM MQ client libraries.

See [Building applications for IBM MQ MQI clients](#) and [Building a procedural application](#) for details on compiling and linking programs that you write in a procedural language.

- The database server can reside on a different machine from the queue manager server, as long as the database client is installed on the same machine as the queue manager, and it supports this function. Consult the database product's documentation to determine whether their client software can be used for two-phase commit systems.

- Although the queue manager behaves as a resource manager (for the purposes of being involved in Scenario 2 global units of work), it is not possible to make one queue manager coordinate another queue manager within its Scenario 1 global units of work.

Multi **Switch load files**

The switch load file is a shared library (a DLL on Windows systems) that is loaded by the code in your IBM MQ application and the queue manager. Its purpose is to simplify the loading of the database's client shared library, and to return the pointers to the XA functions.

The details of the switch load file must be specified before the queue manager is started. The details are placed in the qm.ini file on AIX, Linux, and Windows systems.

- On Windows and Linux (x86 and x86-64 platforms) systems, use the IBM MQ Explorer to update the qm.ini file.
- On all other systems edit the file, qm.ini, directly.

The C source for the switch load file is supplied with the IBM MQ installation if it supports Scenario 1 global units of work. The source contains a function called MQStart. When the switch load file is loaded, the queue manager calls this function, which returns the address of a structure called an *XA switch*.

The XA switch structure exists in the database client shared library, and contains a number of function pointers, as described in [Table 3 on page 55](#):

Function pointer name	XA function	Purpose
xa_open_entry	xa_open	Connect to database
xa_close_entry	xa_close	Disconnect from database
xa_start_entry	xa_start	Start a branch of a global unit of work
xa_end_entry	xa_end	Suspend a branch of a global unit of work
xa_rollback_entry	xa_rollback	Roll back a branch of a global unit of work
xa_prepare_entry	xa_prepare	Prepare to commit a branch of a global unit of work
xa_commit_entry	xa_commit	Commit a branch of a global unit of work
xa_recover_entry	xa_recover	Discover from the database whether it has an in-doubt unit of work
xa_forget_entry	xa_forget	Allow a database to forget a branch of a global unit of work
xa_complete_entry	xa_complete	Complete a branch of a global unit of work

During the first MQBEGIN call in your application, the IBM MQ code that executes as part of MQBEGIN loads the switch load file, and calls the xa_open function in the database shared library. Similarly, during queue manager startup, and on other subsequent occasions, some queue manager processes load the switch load file and call xa_open.

You can reduce the number of xa_* calls by using *dynamic registration*. For a complete description of this optimization technique, see [“XA dynamic registration” on page 74](#).

Configuring your system for database coordination

There are several tasks that you must perform before a database manager can participate in global units of works coordinated by the queue manager. These are described here as follows:

- [“Installing and configuring the database product” on page 56](#)
- [“Creating switch load files” on page 56](#)
- [“Adding configuration information to the queue manager” on page 57](#)
- [“Writing and modifying your applications” on page 59](#)
- [“Testing the system” on page 59](#)

Installing and configuring the database product

To install and configure your database product, see the product's own documentation. This topics in this section describe general configuration issues and how they relate to interoperability between IBM MQ and the database.

Database connections

An application that establishes a standard connection to the queue manager is associated with a thread in a separate local queue manager agent process. (A connection that is not a *fastpath* connection is a *standard* connection in this context. See [Connecting to a queue manager using the MQCONN call.](#))

When the application issues **MQBEGIN**, both it and the agent process call the **xa_open** function in the database client library. In response to this, the database client library code *connects* to the database that is to be involved in the unit of work *from both the application and queue manager processes*. These database connections are maintained as long as the application remains connected to the queue manager.

This is an important consideration if the database supports only a limited number of users or connections, because two connections are being made to the database to support the one application program.

Client/server configuration

The database client library that is loaded into the IBM MQ queue manager and application processes **must** be able to send to and receive from its server. Ensure that:

- The database's client/server configuration files have the correct details
- The relevant environment variables are set in the environment of the queue manager **and** the application processes

Creating switch load files

IBM MQ comes with a sample makefile, used to build switch load files for the supported database managers.

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

The sample makefile, together with all the associated C source files required to build the switch load files, is installed in the following directories:

- For IBM MQ for Windows, in the **MQ_INSTALLATION_PATH\tools\c\samples\xatm** directory
- For IBM MQ for UNIX and Linux systems, in the **MQ_INSTALLATION_PATH/samp/xatm/** directory

The sample targets used to build the switch load files are:

- For Db2, db2swit.
- For Oracle, oraswit
- For Informix, infswit
- For Sybase, sybswit

Windows The generated switch file is placed in C:\Program Files\IBM\MQ\exits.

Linux **UNIX** If you have 32-bit queue managers then the sample make file, xaswit.mak, installs a 32-bit switch load file in /var/mqm/exits.

Linux **UNIX** If you have 64-bit queue managers then the sample make file, xaswit.mak, installs a 32-bit switch load file in /var/mqm/exits, and a 64-bit switch load file in /var/mqm/exits64.

Linux **UNIX** If your system does not support 32-bit compilation, use the 64-bit only target for your database:

- For Db2, db2swit64
- For Oracle, oraswit64
- For Informix, infswit64
- For Sybase, sybswit64

File security

It is possible that your operating system might fail the loading of the switch load file by IBM MQ, for reasons outside the control of IBM MQ. If this occurs, error messages are written to the IBM MQ error logs, and potentially the MQBEGIN call can fail. To help to ensure that your operating system does not fail the loading of the switch load file, you must fulfil the following requirements:

1. The switch load file must be available in the location that is given in the qm.ini file.
2. The switch load file must be accessible to all processes that need to load it, including the queue manager processes and application processes.
3. All of the libraries upon which the switch load file depends, including the libraries that are provided by the database product, must be present and accessible.

Multi *Adding configuration information to the queue manager*

When you have created a switch load file for your database manager, and placed it in a safe location, you must specify that location to your queue manager.

To specify the location, perform the following steps:

- On Windows and Linux (x86 and x86-64 platforms) systems use the IBM MQ Explorer. Specify the details of the switch load file in the queue manager properties panel, under XA resource manager.
- On all other systems specify the details of the switch load file in the XAResourceManager stanza in the queue manager's qm.ini file.

Add an XAResourceManager stanza for the database that your queue manager is going to coordinate. The most common case is for there to be only one database, and therefore only one XAResourceManager stanza. For details of more complicated configurations involving multiple databases, see [“Multiple database configurations”](#) on page 69. The attributes of the XAResourceManager stanza are as follows:

Name=name

User-chosen string that identifies the resource manager. In effect, it gives a name to the XAResourceManager stanza. The name is mandatory and can be up to 31 characters in length.

The name you choose must be unique; there must be only one XAResourceManager stanza with this name in this qm.ini file. The name should also be meaningful, because the queue manager uses it to refer to this resource manager both in queue manager error log messages and in output when the dspmqtrn command is used. (See [“Displaying outstanding units of work with the dspmqtrn command”](#) on page 71 for more information.)

Once you have chosen a name, and have started the queue manager, do not change the Name attribute. For more details about changing configuration information, see [“Changing configuration information”](#) on page 73.

SwitchFile=name

This is the name of the XA switch load file you built earlier. This is a mandatory attribute. The code in the queue manager and IBM MQ application processes tries to load the switch load file on two occasions:

1. At queue manager startup
2. When you make the first call to MQBEGIN in your IBM MQ application process

The security and permissions attributes of your switch load file must allow these processes to perform this action.

XAOpenString=string

This is a string of data that IBM MQ code passes in its calls to the database manager's `xa_open` function. This is an optional attribute; if it is omitted a zero-length string is assumed.

The code in the queue manager and IBM MQ application processes call the `xa_open` function on two occasions:

1. At queue manager startup
2. When you make the first call to MQBEGIN in your IBM MQ application process

The format for this string is particular to each database product, and will be described in the documentation for that product. In general, the `xa_open` string contains authentication information (user name and password) to allow a connection to the database in both the queue manager and the application processes.

From IBM MQ 8.0.0 Fix Pack 4, when the XAOpenString contains a password, you can get IBM MQ to protect this information, rather than having the password visible in plain text in the `qm.ini` file. IBM MQ stores the user name and the password (in an encrypted form) in a different file, and uses these credentials to connect to the database. For details, see [Protection of database authentication details](#).

XACloseString=string

This is a string of data that IBM MQ code passes in its calls to the database manager's `xa_close` function. This is an optional attribute; if it is omitted a zero-length string is assumed.

The code in the queue manager and IBM MQ application processes call the `xa_close` function on two occasions:

1. At queue manager startup
2. When you make a call to MQDISC in your IBM MQ application process, having earlier made a call to MQBEGIN

The format for this string is particular to each database product, and will be described in the documentation for that product. In general, the string is empty, and it is common to omit the XACloseString attribute from the XAResourceManager stanza.

ThreadOfControl=THREAD|PROCESS (default)

The ThreadOfControl value can be THREAD or PROCESS. The queue manager uses it for serialization purposes. This is an optional attribute; if it is omitted, the value PROCESS is assumed.

If the database client code allows threads to call the XA functions without serialization, the value for ThreadOfControl can be THREAD. The queue manager assumes that it can call the XA functions in the database client shared library from multiple threads at the same time, if necessary.

If the database client code does not allow threads to call its XA functions in this way, the value for ThreadOfControl must be PROCESS. In this case, the queue manager serializes all calls to the database client shared library so that only one call at a time is made from within a particular process. You probably also need to ensure that your application performs similar serialization if it runs with multiple threads.

Note that this issue, of the database product's ability to cope with multi-threaded processes in this way, is an issue for that product's vendor. Consult the database product's documentation for details on whether you can set the ThreadOfControl attribute to THREAD or PROCESS. We recommend that,

if you can, you set ThreadOfControl to THREAD. If in doubt, the *safer* option is to set it to PROCESS, although you will lose the potential performance benefits of using THREAD.

Multi *Writing and modifying your applications*

How to implement a global unit of work.

The sample application programs for Scenario 1 global units of work that are supplied with an IBM MQ installation are described in the [“Introducing units of work” on page 51](#).

In general, a global unit of work is implemented in an application by the following method (in pseudocode):

```
MQBEGIN
MQGET
MQPUT
SQL INSERT
MQCMIT
```

The purpose of MQBEGIN is to denote the beginning of a global unit of work. The purpose of MQCMIT is to denote the end of the global unit of work, and to complete it with all participating resource managers, using the two-phase commit protocol.

In between MQBEGIN and MQCMIT, the queue manager does not make any calls to the database to update its resources. That is, the only way a database's tables are changed is by your code (for example, the SQL INSERT in the pseudocode).

The role of the queue manager, as far as the database is concerned, is to tell it when a global unit of work has started, when it has ended, and whether the global unit of work should be committed or rolled-back.

As far as your application is concerned, the queue manager performs two roles: a resource manager (where the resources are messages on queues) and the transaction manager for the global unit of work.

Start with the supplied sample programs, and work through the various IBM MQ and database API calls that are being made in those programs. The API calls concerned are fully documented in [Using the sample procedural programs for Multiplatforms](#), [Data types used in the MQI](#), and (in the case of the database's own API) the database's own documentation.

Multi *Testing the system*

You know whether your application and system are correctly configured only by running them during testing. You can test the system's configuration (the successful communication between queue manager and database) by building and running one of the supplied sample programs.

Multi **Configuring Db2**

Db2 support and configuration information.

The supported levels of Db2 are defined at the [System Requirements for IBM MQ](#) page.

Note: 32-bit instances of Db2 are not supported on platforms where the queue manager is 64-bit.

Do the following:

1. Check the environment variable settings.
2. Create the Db2 switch load file.
3. Add resource manager configuration information.
4. Change Db2 configuration parameters if necessary.

Read this information in conjunction with the general information provided in [“Configuring your system for database coordination” on page 56](#).

Warning: If you run db2profile on AIX and Linux platforms, the environment variable LIBPATH and LD_LIBRARY_PATH are set. It is advisable to unset these environment variables. See [crtmqenv](#) or [setmqenv](#) for more information.

Checking the Db2 environment variable settings

Ensure that your Db2 environment variables are set for queue manager processes **as well as in** your application processes. In particular, you must always set the DB2INSTANCE environment variable **before** you start the queue manager. The DB2INSTANCE environment variable identifies the Db2 instance containing the Db2 databases that are being updated. For example:

- On AIX and Linux systems, use:

```
export DB2INSTANCE=db2inst1
```

- On Windows systems, use:

```
set DB2INSTANCE=Db2
```

On Windows with a Db2 database, you must add the user MUSR_MQADMIN to the DB2USERS group, to enable the queue manager to start.

Creating the Db2 switch load file

The easiest way to create the Db2 switch load file is to use the sample file xaswit.mak, which IBM MQ provides to build the switch load files for a variety of database products.

Windows On Windows systems, you can find xaswit.mak in the directory `MQ_INSTALLATION_PATH\tools\c\samples\xatm`. `MQ_INSTALLATION_PATH` represents the high-level directory in which IBM MQ is installed. To create the Db2 switch load file with Microsoft Visual C++, use:

```
nmake /f xaswit.mak db2swit.dll
```

The generated switch file is placed in `C:\Program Files\IBM\MQ\exits`.

Linux **UNIX** You can find xaswit.mak in the directory `MQ_INSTALLATION_PATH/samp/xatm`. `MQ_INSTALLATION_PATH` represents the high-level directory in which IBM MQ is installed.

Edit xaswit.mak to *uncomment* the lines appropriate to the version of Db2 you are using. Then execute the makefile using the command:

```
make -f xaswit.mak db2swit
```

The generated 32-bit switch load file is placed in `/var/mqm/exits`.

The generated 64-bit switch load file is placed in `/var/mqm/exits64`.

If your system does not support 32-bit compilation, use the 64-bit only target:

```
make -f xaswit.mak db2swit64
```

Adding resource manager configuration information for Db2

You must modify the configuration information for the queue manager to declare Db2 as a participant in global units of work. Modifying configuration information in this way is described in more details in [“Adding configuration information to the queue manager” on page 57](#).

- On Windows and Linux (x86 and x86-64 platforms) systems, use the IBM MQ Explorer. Specify the details of the switch load file in the queue manager properties panel, under XA resource manager.
- On all other systems specify the details of the switch load file in the XAResourceManager stanza in the queue manager's qm.ini file.

Figure 6 on page 61 is a UNIX sample, showing an XAResourceManager entry where the database to be coordinated is called mydbname, this name being specified in the XAOpenString:

```
XAResourceManager:
  Name=mydb2
  SwitchFile=db2swit
  XAOpenString=mydbname,myuser,mypasswd,toc=t
  ThreadOfControl=THREAD
```

Figure 6. Sample XAResourceManager entry for Db2 on UNIX

Note:

1. ThreadOfControl=THREAD cannot be used with Db2 versions earlier than 8. Set ThreadOfControl and the XAOpenString parameter toc to one of the following combinations:
 - ThreadOfControl=THREAD and toc=t
 - ThreadOfControl=PROCESS and toc=p

If you are using the jdbcdB2 XA switch load file to enable JDBC/JTA coordination, you must use ThreadOfControl=PROCESS and toc=p.

Changing Db2 configuration parameters

For each Db2 database that the queue manager is coordinating, you must set database privileges, change the tp_mon_name parameter, and reset the maxappls parameter. To do this, perform the following steps:

Set database privileges

The queue manager processes run with effective user and group mqm on AIX and Linux systems. On Windows systems, they run as the user that started the queue manager. This can be one of:

1. The user who issued the stmqm command, or
2. The user under which the IBM MQ Service COM server runs

By default, this user is called MUSR_MQADMIN.

If you have not specified a user name and password on the xa_open string, **the user under which the queue manager is running** is used by Db2 to authenticate the xa_open call. If this user (for example, user mqm on AIX and Linux systems) does not have minimal privileges in the database, the database refuses to authenticate the xa_open call.

The same considerations apply to your application process. If you have not specified a user name and password on the xa_open string, the user under which your application is running is used by Db2 to authenticate the xa_open call that is made during the first MQBEGIN. Again, this user must have minimal privileges in the database for this to work.

For example, give the mqm user connect authority in the mydbname database by issuing the following Db2 commands:

```
db2 connect to mydbname
db2 grant connect on database to user mqm
```

See [“Security considerations” on page 69](#) for more information about security.

Windows Change the TP_MON_NAME parameter

For Db2 on Windows systems only, change the TP_MON_NAME configuration parameter to name the DLL that Db2 uses to call the queue manager for dynamic registration.

Use the command `db2 update dbm cfg using TP_MON_NAME mqmax` to name MQMAX.DLL as the library that Db2 uses to call the queue manager. This must be present in a directory within PATH.

Reset the maxappls parameter

You might need to review your setting for the *maxappls* parameter, which limits the maximum number of applications that can be connected to a database. Refer to [“Installing and configuring the database product”](#) on page 56.

Multi **Configuring Oracle**

Oracle support and configuration information.

Complete the following steps:

1. Check environment variable settings.
2. Create the Oracle switch load file.
3. Add resource manager configuration information.
4. Change the Oracle configuration parameters, if necessary.

For a current list of levels of Oracle supported by IBM MQ, see [System Requirements for IBM MQ](#).

Checking the Oracle environment variable settings

Ensure that your Oracle environment variables are set for queue manager processes as well as in your application processes. In particular, always set the following environment variables before starting the queue manager:

ORACLE_HOME

The Oracle home directory. For example, on AIX and Linux systems, use:

```
export ORACLE_HOME=/opt/oracle/product/8.1.6
```

On Windows systems, use:

```
set ORACLE_HOME=c:\oracle\ora81
```

ORACLE_SID

The Oracle SID being used. If you are using Net8 for client/server connectivity, you might not have to set this environment variable. Consult your Oracle documentation.

The subsequent example is an example of setting this environment variable, on AIX and Linux systems:

```
export ORACLE_SID=sid1
```

The equivalent on Windows systems is:

```
set ORACLE_SID=sid1
```

Note: The PATH environment variable must be set to include the binaries directory (for example ORACLE_INSTALL_DIR/VERSION/32BIT_NAME/bin or ORACLE_INSTALL_DIR/VERSION/64BIT_NAME/bin), otherwise you might see a message stating that oraclient libraries are missing from the machine.

If you run queue managers on Windows 64 bit systems, then only 64 bit Oracle clients must be installed. The switch load file, loaded by 64 bit queue managers, must access the Oracle 64 bit client libraries.

Creating the Oracle switch load file

To create the Oracle switch load file, use the sample file `xaswit.mak`, which IBM MQ provides to build the switch load files for various database products.

Windows On Windows systems, you can find `xaswit.mak` in the directory `C:\Program Files\IBM\MQ\tools\c\samples\xatm`. To create the Oracle switch load file with Microsoft Visual C++, use:

```
nmake /f xaswit.mak oraswit.dll
```

Note: These switch load files can be used only with C applications. For Java applications, see [JTA/JDBC coordination using IBM MQ classes for Java](#).

The generated switch file is placed in `MQ_INSTALLATION_PATH\exits`. `MQ_INSTALLATION_PATH` represents the high-level directory in which IBM MQ is installed.

Linux **UNIX** You can find `xaswit.mak` in the directory `MQ_INSTALLATION_PATH/samp/xatm`. `MQ_INSTALLATION_PATH` represents the high-level directory in which IBM MQ is installed.

Edit `xaswit.mak` to uncomment the lines appropriate to the version of Oracle you are using. Then execute the makefile using the command:

```
make -f xaswit.mak oraswit
```

The contents of `MQ_INSTALLATION_PATH/samp/xatm` are read-only when IBM MQ is installed, so to edit `xaswit.mak`, copy all the files out of `samp/xatm` to another directory, modify `xaswit.mak`, and then run `make -f xaswit.mak oraswit` from that directory.

The generated 32 bit switch load file is placed in `/var/mqm/exits`.

The generated 64 bit switch load file is placed in `/var/mqm/exits64`.

If your system does not support 32-bit compilation, use the 64-bit only target:

```
make -f xaswit.mak oraswit64
```

Adding resource manager configuration information for Oracle

You must modify the configuration information for the queue manager to declare Oracle as a participant in global units of work. Modifying the configuration information for the queue manager in this way is described in more detail in [“Adding configuration information to the queue manager”](#) on page 57.

- On Windows and Linux (x86 and x86-64 platforms) systems, use the IBM MQ Explorer. Specify the details of the switch load file in the queue manager properties panel, under XA resource manager.
- On all other systems specify the details of the switch load file in the `XAResourceManager` stanza in the `qm.ini` file of the queue manager.

Figure 7 on page 63 is an AIX and Linux systems sample showing an `XAResourceManager` entry. You must add a `LogDir` to the XA open string so that all error and tracing information is logged to the same place.

```
XAResourceManager:
  Name=myoracle
  SwitchFile=oraswit
  XAOpenString=Oracle_XA+Acc=P/myuser/mypasswd+SesTm=35+LogDir=/tmp+threads=true
  ThreadOfControl=THREAD
```

Figure 7. Sample `XAResourceManager` entry for Oracle on AIX and Linux platforms

Note:

1. In Figure 7 on page 63, the `xa_open` string has been used with four parameters. Additional parameters can be included as described in Oracle's documentation.

2. When using the IBM MQ parameter `ThreadOfControl=THREAD` you must use the Oracle parameter `+threads=true` in the `XAResourceManager` stanza.

See the *Oracle8 Server Application Developer's Guide* for more information about the `xa_open` string.

Changing Oracle configuration parameters

For each Oracle database that the queue manager is coordinating, you must review your maximum sessions and set database privileges. To do so, complete these steps:

Review your maximum sessions

You might have to review your `LICENSE_MAX_SESSIONS` and `PROCESSES` settings to take into account the additional connections required by processes belonging to the queue manager. See [“Installing and configuring the database product” on page 56](#) for more details.

Set database privileges

The Oracle user name specified in the `xa_open` string must have privileges to access the `DBA_PENDING_TRANSACTIONS` view, as described in the Oracle documentation.

The necessary privilege can be given using the following example command:

```
grant select on DBA_PENDING_TRANSACTIONS to myuser;
```

Configuring Informix

Informix support and configuration information.

Complete the following steps:

1. Ensure that you have installed the appropriate Informix client SDK:
 - 32 bit queue managers and applications require a 32 bit Informix client SDK.
 - 64 bit queue managers and applications require a 64 bit Informix client SDK.
2. Ensure that Informix databases are created correctly.
3. Check environment variable settings.
4. Build the Informix switch load file.
5. Add resource manager configuration information.

For a current list of levels of Informix supported by IBM MQ, see [System Requirements for IBM MQ](#).

Ensuring that Informix databases are created correctly

Every Informix database that is to be coordinated by an IBM MQ queue manager must be created specifying the **log** parameter. For example:

```
create database mydbname with log;
```

IBM MQ queue managers are unable to coordinate Informix databases that do not have the `log` parameter specified on creation. If a queue manager attempts to coordinate an Informix database that does not have the `log` parameter specified on creation, the `xa_open` call to Informix fails, and a number of FFST errors are generated.

Checking the Informix environment variable settings

Ensure that your Informix environment variables are set for queue manager processes **as well as in** your application processes. In particular, always set the following environment variables **before** starting the queue manager:

INFORMIXDIR

The directory of the Informix product installation.

- For 32 bit AIX and Linux applications, use the following command:

```
export INFORMIXDIR=/opt/informix/32-bit
```

- For 64 bit AIX and Linux applications, use the following command:

```
export INFORMIXDIR=/opt/informix/64-bit
```

- For Windows applications, use the following command:

```
set INFORMIXDIR=c:\informix
```

For systems that have 64 bit queue managers that must support both 32 bit and 64 bit applications, you need both the Informix 32 bit and 64 bit client SDKs installed. The sample makefile `xaswit.mak`, used for creating a switch load file also sets both product installation directories.

INFORMIXSERVER

The name of the Informix server. For example, on AIX and Linux systems, use:

```
export INFORMIXSERVER=hostname_1
```

On Windows systems, use:

```
set INFORMIXSERVER=hostname_1
```

ONCONFIG

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

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

On Windows systems, use:

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

Creating the Informix switch load file

To create the Informix switch load file, use the sample file `xaswit.mak`, which IBM MQ provides to build the switch load files for various database products.

Windows On Windows systems, you can find `xaswit.mak` in the directory `MQ_INSTALLATION_PATH\tools\c\samples\xa\atm`. `MQ_INSTALLATION_PATH` represents the high-level directory in which IBM MQ is installed. To create the Informix switch load file with Microsoft Visual C++, use:

```
nmake /f xaswit.mak infswit.dll
```

The generated switch file is placed in `C:\Program Files\IBM\MQ\exits`.

Linux **UNIX** You can find `xaswit.mak` in the directory `MQ_INSTALLATION_PATH/samp/xatm`. `MQ_INSTALLATION_PATH` represents the high-level directory in which IBM MQ is installed.

Edit `xaswit.mak` to *uncomment* the lines appropriate to the version of Informix you are using. Then execute the makefile using the command:

```
make -f xaswit.mak infswit
```

The generated 32 bit switch load file is placed in /var/mqm/exits.

The generated 64 bit switch load file is placed in /var/mqm/exits64.

If your system does not support 32-bit compilation, use the 64-bit only target:

```
make -f xaswit.mak infswit64
```

Adding resource manager configuration information for Informix

You must modify the configuration information for the queue manager to declare Informix as a participant in global units of work. Modifying the configuration information for the queue manager in this way is described in more detail in [“Adding configuration information to the queue manager”](#) on page 57.

- On Windows and Linux (x86 and x86-64 platforms) systems, use the IBM MQ Explorer. Specify the details of the switch load file in the queue manager properties panel, under XA resource manager.
- On all other systems specify the details of the switch load file in the XAResourceManager stanza in the qm.ini file of the queue manager.

Figure 8 on page 66 is a UNIX sample, showing a qm.ini XAResourceManager entry where the database to be coordinated is called mydbname, this name being specified in the XAOpenString:

```
XAResourceManager:
  Name=myinformix
  SwitchFile=infswit
  XAOpenString=DB=mydbname@myinformixserver\;USER=myuser\;PASSWD=mypasswd
  ThreadOfControl=THREAD
```

Figure 8. Sample XAResourceManager entry for Informix on UNIX

Note: By default the sample xaswit.mak on UNIX creates a switch load file that uses threaded Informix libraries. You must ensure that ThreadOfControl is set to THREAD when using these Informix libraries. In Figure 8 on page 66, the qm.ini file XAResourceManager stanza attribute ThreadOfControl is set to THREAD. When THREAD is specified, applications must be built using the threaded Informix libraries and the IBM MQ threaded API libraries.

The XAOpenString attribute must contain the database name, followed by the @ symbol, and then followed by the Informix server name.

To use the nonthreaded Informix libraries, you must ensure that the qm.ini file XAResourceManager stanza attribute ThreadOfControl is set to PROCESS. You must also make the following changes to the sample xaswit.mak:

1. Uncomment the generation of a nonthreaded switch load file.
2. Comment out the generation of the threaded switch load file.

Sybase configuration

Sybase support and configuration information.

Complete the following steps:

1. Ensure you have installed the Sybase XA libraries, for example by installing the XA DTM option.
2. Check environment variable settings.
3. Enable Sybase XA support.
4. Create the Sybase switch load file.
5. Add resource manager configuration information.

For a current list of levels of Sybase supported by IBM MQ, see [System Requirements for IBM MQ](#).

Checking the Sybase environment variable settings

Ensure that your Sybase environment variables are set for queue manager processes as well as in your application processes. In particular, always set the following environment variables before starting the queue manager:

SYBASE

The location of the Sybase product installation. For example, on AIX and Linux systems, use:

```
export SYBASE=/sybase
```



On Windows systems, use:

```
set SYBASE=c:\sybase
```

SYBASE_OCS

The directory under SYBASE where you have installed the Sybase client files.

For example, on AIX and Linux systems, use:

```
export SYBASE_OCS=OCS-12_0
```



On Windows systems, use:

```
set SYBASE_OCS=OCS-12_0
```

Enabling Sybase XA support

Within the Sybase XA configuration file `$SYBASE/$SYBASE_OCS/xa_config`, define a Logical Resource Manager (LRM) for each connection to the Sybase server that is being updated. Here is an example of the contents of `$SYBASE/$SYBASE_OCS/xa_config`:

```
# The first line must always be a comment

[xa]

LRM=lrnname
server=servername
```

Creating the Sybase switch load file

To create the Sybase switch load file, use the sample files supplied with IBM MQ.



On Windows systems, you can find `xaswit.mak` in the directory `C:\Program Files\IBM\MQ\tools\c\samples\xatm`. To create the Sybase switch load file with Microsoft Visual C++, use:

```
nmake /f xaswit.mak sybswit.dll
```

The generated switch file is placed in `C:\Program Files\IBM\MQ\exits`.

You can find `xaswit.mak` in the directory `MQ_INSTALLATION_PATH/samp/xatm`. `MQ_INSTALLATION_PATH` represents the high-level directory in which IBM MQ is installed.

Edit `xaswit.mak` to *uncomment* the lines appropriate to the version of Sybase you are using. Then execute the makefile using the command:


```
make -f xaswit.mak sybswit
```

The generated 32-bit switch load file is placed in `/var/mqm/exits`.

The generated 64-bit switch load file is placed in `/var/mqm/exits64`.

If your system does not support 32-bit compilation, use the 64-bit only target:

```
make -f xaswit.mak sybswit64
```

Note:  On AIX, the sample makefile has been modified as shown in the following example so that you can select a different SYBLINKFLAG64 value, depending on whether you are using Sybase 15 ESD#5 or later, or an earlier version of Sybase.

```
SYBLINKFLAGS32=-btrl  
# The following line is for Sybase 15  
#SYBLINKFLAG64=-btrl  
# The following line is for Sybase 16  
SYBLINKFLAG64=-bstatic -bdynamic
```



The only change you need to make to the makefile is to ensure that only one of the SYBLINKFLAG64 values is uncommented. The default is Sybase 16, which is the value to use for 15 #ESD5 and later.



Any XA switch file that is produced is linked to that specific release of Sybase and must not be moved to other platforms.

If the level of Sybase is changed, then the XA switch file should be rebuilt.

Adding resource manager configuration information for Sybase

You must modify the configuration information for the queue manager to declare Sybase as a participant in global units of work. Modifying the configuration information is described in more detail in [“Adding configuration information to the queue manager”](#) on page 57.

-   On Windows and Linux (x86 and x86-64 platforms) systems, use the IBM MQ Explorer. Specify the details of the switch load file in the queue manager properties panel, under XA resource manager.
- On all other systems specify the details of the switch load file in the XAResourceManager stanza in the queue manager's `qm.ini` file.

  The following example shows a Sample XAResourceManager entry for Sybase on AIX and Linux, which uses the database associated with the `lrnmname` LRM definition in the Sybase XA configuration file `$SYBASE/$SYBASE_OCS/xa_config`. Include a log file name if you want XA function calls to be logged:

```
XAResourceManager:  
  Name=mysybase  
  SwitchFile=sybswit  
  XAOpenString=-Uuser -Ppassword -Nlrnmname -L/tmp/sybase.log -Txa  
  ThreadOfControl=THREAD
```

Using multi-threaded programs with Sybase

If you are using multi-threaded programs with IBM MQ global units of work incorporating updates to Sybase, you must use the value `THREAD` for the `ThreadOfControl` parameter. Also ensure that you link your program (and the switch load file) with the threadsafe Sybase libraries (the `_r` versions). Using the value `THREAD` for the `ThreadOfControl` parameter is shown in the previous example.

Multi Multiple database configurations

If you want to configure the queue manager so that updates to multiple databases can be included within global units of work, add an XAResourceManager stanza for each database.

If the databases are all managed by the same database manager, each stanza defines a separate database. Each stanza specifies the same *SwitchFile*, but the contents of the *XAOpenString* are different because it specifies the name of the database being updated. For example, the stanzas shown in [Figure 9](#) on [page 69](#) configure the queue manager with the Db2 databases *MQBankDB* and *MQFeeDB* on AIX and Linux systems.

Important: You cannot have multiple stanzas pointing to the same database. This configuration does not work under any circumstances, and if you try this configuration it fails.

You will receive errors of the form when the MQ code makes its second `xa_open` call in any process in this environment, the database software fails the second `xa_open` with a -5 error, `XAER_INVAL`.

```
XAResourceManager:
Name=DB2 MQBankDB
SwitchFile=db2swit
XAOpenString=MQBankDB

XAResourceManager:
Name=DB2 MQFeeDB
SwitchFile=db2swit
XAOpenString=MQFeeDB
```

Figure 9. Sample XAResourceManager entries for multiple Db2 databases

If the databases to be updated are managed by different database managers, add an XAResourceManager stanza for each. In this case, each stanza specifies a different *SwitchFile*. For example, if *MQFeeDB* is managed by Oracle instead of Db2, use the following stanzas on AIX and Linux systems:

```
XAResourceManager:
Name=DB2 MQBankDB
SwitchFile=db2swit
XAOpenString=MQBankDB

XAResourceManager:
Name=Oracle MQFeeDB
SwitchFile=oraswit
XAOpenString=Oracle_XA+Acc=P/myuser/mypassword+SesTm=35+LogDir=/tmp/ora.log+DB=MQFeeDB
```

Figure 10. Sample XAResourceManager entries for a Db2 and Oracle database

In principle, there is no limit to the number of database instances that can be configured with a single queue manager.

Note: For information on support for including Informix databases in multiple database updates within global units of work, check the product readme file.

Multi Security considerations

Considerations for running your database under the XA model.

The following information is provided for guidance only. In all cases, refer to the documentation provided with the database manager to determine the security implications of running your database under the XA model.

An application process denotes the start of a global unit of work using the MQBEGIN verb. The first MQBEGIN call that an application issues connects to all participating databases by calling their client library code at the xa_open entry point. All the database managers provide a mechanism for supplying a user ID and password in their XAOpenString. This is the only time that authentication information flows.

Note that, on AIX and Linux platforms, fastpath applications must run with an effective user ID of mqm while making MQI calls.

Considerations when contact is lost with the XA resource manager

The queue manager tolerates database managers not being available. This means you can start and stop the queue manager independently from the database server. When contact is restored, the queue manager and database resynchronize. You can also use the rsvmqtrn command to manually resolve in-doubt units of work.

In normal operations, only a minimal amount of administration is necessary after you have completed the configuration steps. The administration job is made easier because the queue manager tolerates database managers not being available. In particular this means that:

- The queue manager can start at any time without first starting each of the database managers.
- The queue manager does not need to stop and restart if one of the database managers becomes unavailable.

This allows you to start and stop the queue manager independently from the database server.

Whenever contact is lost between the queue manager and a database, they need to resynchronize when both become available again. Resynchronization is the process by which any in-doubt units of work involving that database are completed. In general, this occurs automatically without the need for user intervention. The queue manager asks the database for a list of units of work that are in doubt. It then instructs the database to either commit or roll back each of these in-doubt units of work.

When a queue manager starts, it resynchronizes with each database. When an individual database becomes unavailable, only that database needs to be resynchronized the next time that the queue manager notices it is available again.

The queue manager regains contact with a previously unavailable database automatically as new global units of work are started with MQBEGIN. It does this by calling the xa_open function in the database client library. If this xa_open call fails, MQBEGIN returns with a completion code of MQCC_WARNING and a reason code of MQRC_PARTICIPANT_NOT_AVAILABLE. You can retry the MQBEGIN call later.

Do not continue to attempt a global unit of work that involves updates to a database that has indicated failure during MQBEGIN. There will not be a connection to that database through which updates can be made. Your only options are to end the program, or to retry MQBEGIN periodically in the hope that the database might become available again.

Alternatively, you can use the rsvmqtrn command to resolve explicitly all in-doubt units of work.

In-doubt units of work

A database might have in-doubt units of work if contact with the queue manager is lost after the database manager has been instructed to prepare. Until the database server receives the outcome from the queue manager (commit or roll back), it needs to retain the database locks associated with the updates.

Because these locks prevent other applications from updating or reading database records, resynchronization needs to take place as soon as possible.

If, for some reason, you cannot wait for the queue manager to resynchronize with the database automatically, you can use facilities provided by the database manager to commit or roll back the database updates manually. In the *X/Open Distributed Transaction Processing: The XA Specification*, this is called making a *heuristic* decision. Use it only as a last resort because of the possibility of compromising data integrity; you might, for example, mistakenly roll back the database updates when all other participants have committed their updates.

It is far better to restart the queue manager, or use the `rsvmqtrn` command when the database has been restarted, to initiate automatic resynchronization.

Displaying outstanding units of work with the `dspmqtrn` command

You can use the `dspmqtrn` command with the `-i` parameter to display internally originated indoubt transactions.

While a database manager is unavailable, you can use the `dspmqtrn` command to check the state of outstanding global units of work involving that database.

The `dspmqtrn` command displays only those units of work in which one or more participants are in doubt. The participants are awaiting the decision from the queue manager to commit or roll back the prepared updates.

For each of these global units of work, the state of each participant is displayed in the output from `dspmqtrn`. If the unit of work did not update the resources of a particular resource manager, it is not displayed.

With respect to an in-doubt unit of work, a resource manager is said to have done one of the following things:

Prepared

The resource manager is prepared to commit its updates.

Committed

The resource manager has committed its updates.

Rolled-back

The resource manager has rolled back its updates.

Participated

The resource manager is a participant, but has not prepared, committed, or rolled back its updates.

When the queue manager is restarted, it asks each database having an XAResourceManager stanza for a list of its in-doubt global units of work. If the database has not been restarted, or is otherwise unavailable, the queue manager cannot yet deliver to the database the final outcomes for those units of work. The outcome of the in-doubt units of work is delivered to the database at the first opportunity when the database is again available.

In this case, the database manager is reported as being in the prepared state until resynchronization has occurred.

Whenever the `dspmqtrn` command displays an in-doubt unit of work, it first lists all the possible resource managers that might be participating. These are allocated a unique identifier, RMId, which is used instead of the *Name* of the resource managers when reporting their state with respect to an in-doubt unit of work.

[Sample dspmqtrn output](#) shows the result of issuing the following command:

```
dspmqtrn -m MY_QMGR
```

```

AMQ7107: Resource manager 0 is MQSeries.
AMQ7107: Resource manager 1 is DB2 MQBankDB.
AMQ7107: Resource manager 2 is DB2 MQFeedB.

AMQ7056: Transaction number 0,1.
      XID: formatID 5067085, gtrid_length 12, bqual_length 4
           gtrid [3291A5060000201374657374]
           bqual [00000001]
AMQ7105: Resource manager 0 has committed.
AMQ7104: Resource manager 1 has prepared.
AMQ7104: Resource manager 2 has prepared.

```

where *Transaction number* is the ID of the transaction which can be used with the **rsvmqtrn** command. See [AMQ7xxx: IBM MQ product messages](#) for further information. The *XID* variables are part of the *X/Open XA Specification* ; for the most up-to-date information about this specification see: <https://publications.opengroup.org/c193>.

Figure 11. Sample *dspmqtrn* output

The output in [Sample dspmqtrn output](#) shows that there are three resource managers associated with the queue manager. The first is resource manager 0, which is the queue manager itself. The other two resource manager instances are the MQBankDB and MQFeedB Db2 databases.

The example shows only a single in-doubt unit of work. A message is issued for all three resource managers, which means that updates were made to the queue manager and both Db2 databases within the unit of work.

The updates made to the queue manager, resource manager 0, have been committed. The updates to the Db2 databases are in the prepared state, which means that Db2 must have become unavailable before it was called to commit the updates to the MQBankDB and MQFeedB databases.

The in-doubt unit of work has an external identifier called an XID (*transaction id*). This is a piece of data given to Db2 by the queue manager to identify its portion of the global unit of work.

See [dspmqtrn](#) for more information.

Resolving outstanding units of work with the rsvmqtrn command

Outstanding units of work complete when the queue manager and Db2 resynchronize.

The output shown in [Figure 11 on page 72](#) shows a single in-doubt unit of work in which the commit decision has yet to be delivered to both Db2 databases.

To complete this unit of work, the queue manager and Db2 need to resynchronize when Db2 next becomes available. The queue manager uses the start of new units of work as an opportunity to regain contact with Db2. Alternatively, you can instruct the queue manager to resynchronize explicitly using the **rsvmqtrn** command.

Do this soon after Db2 has been restarted, so that any database locks associated with the in-doubt unit of work are released as quickly as possible. Use the -a option, which tells the queue manager to resolve all in-doubt units of work. In the following example, Db2 has restarted, so the queue manager can resolve the in-doubt unit of work:

```

> rsvmqtrn -m MY_QMGR -a
Any in-doubt transactions have been resolved.

```

Mixed outcomes and errors

Although the queue manager uses a two-phase commit protocol, this does not completely remove the possibility of some units of work completing with mixed outcomes. This is where some participants commit their updates and some back out their updates.

Units of work that complete with a mixed outcome have serious implications because shared resources that should have been updated as a single unit of work are no longer in a consistent state.

Mixed outcomes are mainly caused when heuristic decisions are made about units of work instead of allowing the queue manager to resolve in-doubt units of work itself. Such decisions are outside the queue manager's control.

Whenever the queue manager detects a mixed outcome, it produces FFST information and documents the failure in its error logs, with one of two messages:

- If a database manager rolls back instead of committing:

```
AMQ7606 A transaction has been committed but one or more resource
managers have rolled back.
```

- If a database manager commits instead of rolling back:

```
AMQ7607 A transaction has been rolled back but one or more resource
managers have committed.
```

Further messages identify the databases that are heuristically damaged. It is then your responsibility to locally restore consistency to the affected databases. This is a complicated procedure in which you need first to isolate the update that has been wrongly committed or rolled back, then to undo or redo the database change manually.

Changing configuration information

After the queue manager has successfully started to coordinate global units of work, do not change any of the resource manager configuration information.

If you need to change the configuration information you can do so at any time, but the changes do not take effect until after the queue manager has been restarted.

If you remove the resource manager configuration information for a database, you are effectively removing the ability for the queue manager to contact that database manager.

Never change the *Name* attribute in any of your resource manager configuration information. This attribute uniquely identifies that database manager instance to the queue manager. If you change this unique identifier, the queue manager assumes that the database has been removed and a completely new instance has been added. The queue manager still associates outstanding units of work with the old *Name*, possibly leaving the database in an in-doubt state.

Removing database manager instances

If you need to remove a database from your configuration permanently, ensure that the database is not in doubt before you restart the queue manager.

Database products provide commands for listing in-doubt transactions. If there are any in-doubt transactions, first allow the queue manager to resynchronize with the database. Do this by starting the queue manager. You can verify that resynchronization has taken place by using the **rsvmqtrn** command or the database's own command for viewing in-doubt units of work. Once you are satisfied that resynchronization has taken place, end the queue manager and remove the database's configuration information.

If you fail to observe this procedure the queue manager still remembers all in-doubt units of work involving that database. A warning message, AMQ7623, is issued every time the queue manager is restarted. If you are never going to configure this database with the queue manager again, use the **-r** option of the **rsvmqtrn** command to instruct the queue manager to forget about the database's participation in its in-doubt transactions. The queue manager forgets about such transactions only when in-doubt transactions have been completed with all participants.

There are times when you might need to remove some resource manager configuration information temporarily. On AIX and Linux systems this is best achieved by commenting out the stanza so that it can be easily reinstated at a later time. You might decide to do this if there are errors every time the queue manager contacts a particular database or database manager. Temporarily removing the resource manager configuration information concerned allows the queue manager to start global units of work

involving all the other participants. Here is an example of a commented-out XAResourceManager stanza follows:

```
# This database has been temporarily removed
#XAResourceManager:
#  Name=mydb2
#  SwitchFile=db2swit
#  XAOpenString=mydbname,myuser,mypassword,toc=t
#  ThreadOfControl=THREAD
```

Figure 12. Commented-out XAResourceManager stanza on AIX and Linux systems

On Windows systems, use the IBM MQ Explorer to delete the information about the database manager instance. Take great care to type in the correct name in the *Name* field when reinstating it. If you mistype the name, you may face in-doubt problems, as described in [“Changing configuration information”](#) on page 73.

XA dynamic registration

The XA specification provides a way of reducing the number of `xa_*` calls that a transaction manager makes to a resource manager. This optimization is known as *dynamic registration*.

Dynamic registration is supported by Db2. Other databases might support it; consult the documentation for your database product for details.

Why is the dynamic registration optimization useful? In your application, some global units of work might contain updates to database tables; others might not contain such updates. When no persistent update has been made to a database's tables, there is no need to include that database in the commit protocol that occurs during MQCMIT.

Whether or not your database supports dynamic registration, your application calls `xa_open` during the first MQBEGIN call on an IBM MQ connection. It also calls `xa_close` on the subsequent MQDISC call. The pattern of subsequent XA calls depends on whether the database supports dynamic registration:

If your database does not support dynamic registration...

Every global unit of work involves several XA function calls made by IBM MQ code into the database client library, regardless of whether you made a persistent update to the tables of that database within your unit of work. These include:

- `xa_start` and `xa_end` from the application process. These are used to declare the beginning and end of a global unit of work.
- `xa_prepare`, `xa_commit`, and `xa_rollback` from the queue manager agent process, `amqzlaa0`. These are used to deliver the outcome of the global unit of work: the commit or rollback decision.

In addition, the queue manager agent process also calls `xa_open` during the first MQBEGIN.

If your database supports dynamic registration...

The IBM MQ code makes only those XA function calls that are necessary. For a global unit of work that has **not** involved persistent updates to database resources, there are **no** XA calls to the database. For a global unit of work that **has** involved such persistent updates, the calls are to:

- `xa_end` from the application process to declare the end of the global unit of work.
- `xa_prepare`, `xa_commit`, and `xa_rollback` from the queue manager agent process, `amqzlaa0`. These are used to deliver the outcome of the global unit of work: the commit or rollback decision.

For dynamic registration to work, it is vital that the database has a way of telling IBM MQ when it has performed a persistent update that it wants to be included in the current global unit of work. IBM MQ provides the `ax_reg` function for this purpose.

The database's client code that runs in your application process finds the `ax_reg` function and calls it, to *dynamically register* the fact it has done persistent work within the current global unit of work.

In response to this `ax_reg` call, IBM MQ records that the database has participated. If this is the first `ax_reg` call on this IBM MQ connection, the queue manager agent process calls `xa_open`.

The database client code make this `ax_reg` call when it is running in your process, for example, during an SQL UPDATE call or whatever call in the database's client API is responsible

Multi Error conditions

In XA dynamic registration there is a possibility of a confusing failure in the queue manager.

A common example is if you forget to set your database environment variables properly before starting your queue manager, the queue manager's calls to `xa_open` fail. No global units of work can be used.

To avoid this, ensure that you have set the relevant environment variables before starting the queue manager. Review your database product's documentation, and the advice given in [“Configuring Db2” on page 59](#), [“Configuring Oracle” on page 62](#), and [“Sybase configuration” on page 66](#).

With all database products, the queue manager calls `xa_open` once at queue manager startup, as part of the recovery session (as explained in [“Considerations when contact is lost with the XA resource manager” on page 70](#)). This `xa_open` call fails if you set your database environment variables incorrectly, but it does not cause the queue manager to fail to start. This is because the same `xa_open` error code is used by the database client library to indicate that the database server is unavailable. IBM MQ does not treat this as a serious error, as the queue manager must be able to start to continue processing data outside global units of work involving that database.

Subsequent calls to `xa_open` are made from the queue manager during the first MQBEGIN on an IBM MQ connection (if dynamic registration is not being used) or during a call by the database client code to the IBM MQ-provided `ax_reg` function (if dynamic registration is being used).

The **timing** of any error conditions (or, occasionally, FFST reports) depends on whether you are using dynamic registration:

- If you are using dynamic registration, your MQBEGIN call could succeed, but your SQL UPDATE (or similar) database call will fail.
- If you are not using dynamic registration, your MQBEGIN call will fail.

Ensure that your environment variables are set correctly in your application and queue manager processes.

Multi Summarizing XA calls

Here is a list of the calls that are made to the XA functions in a database client library as a result of the various MQI calls that control global units of work. This is not a complete description of the protocol described in the XA specification; it is provided as a brief overview.

Note that `xa_start` and `xa_end` calls are always called by IBM MQ code in the application process, whereas `xa_prepare`, `xa_commit`, and `xa_rollback` are always called from the queue manager agent process, `amqzlaa0`.

The `xa_open` and `xa_close` calls shown in this table are all made from the application process. The queue manager agent process calls `xa_open` in the circumstances described in [“Error conditions” on page 75](#).

Table 4. Summary of XA function calls		
MQI call	XA calls made with dynamic registration	XA calls made without dynamic registration
First MQBEGIN	<code>xa_open</code>	<code>xa_open</code> <code>xa_start</code>
Subsequent MQBEGIN	No XA calls	<code>xa_start</code>

Table 4. Summary of XA function calls (continued)

MQI call	XA calls made with dynamic registration	XA calls made without dynamic registration
MQCMIT (without ax_reg being called during the current global unit of work)	No XA calls	xa_end xa_prepare xa_commit xa_rollback
MQCMIT (with ax_reg being called during the current global unit of work)	xa_end xa_prepare xa_commit xa_rollback	Not applicable. No calls are made to ax_reg in non-dynamic mode.
MQBACK (without ax_reg being called during the current global unit of work)	No XA calls	xa_end xa_rollback
MQBACK (with ax_reg being called during the current global unit of work)	xa_end xa_rollback	Not applicable. No calls are made to ax_reg in non-dynamic mode.
MQDISC, where MQCMIT or MQBACK was called first. If they were not, MQCMIT processing is first done during MQDISC.	xa_close	xa_close

Notes:

1. For MQCMIT, xa_commit is called if xa_prepare is successful. Otherwise, xa_rollback is called.

Scenario 2: Other software provides the coordination

In scenario 2, an external transaction manager coordinates global units of work, starting and committing them under control of the transaction manager's API. The MQBEGIN, MQCMIT, and MQBACK verbs are unavailable.

This section describes this scenario, including:

- [“External sync point coordination” on page 76](#)
- [“Using CICS” on page 79](#)
- [“Using the Microsoft Transaction Server \(COM+\)” on page 83](#)

External sync point coordination

A global unit of work can also be coordinated by an external X/Open XA-compliant transaction manager. Here the IBM MQ queue manager participates in, but does not coordinate, the unit of work.

The flow of control in a global unit of work coordinated by an external transaction manager is as follows:

1. An application tells the external sync point coordinator (for example, TXSeries) that it wants to start a transaction.
2. The sync point coordinator tells known resource managers, such as IBM MQ, about the current transaction.
3. The application issues calls to resource managers associated with the current transaction. For example, the application could issue MQGET calls to IBM MQ.
4. The application issues a commit or backout request to the external sync point coordinator.

5. The sync point coordinator completes the transaction by issuing the appropriate calls to each resource manager, typically using two-phase commit protocols.

The supported levels of external sync point coordinators that can provide a two-phase commit process for transactions in which IBM MQ participates are defined at [System Requirements for IBM MQ](#).

The rest of this section describes how to enable external units of work.

The IBM MQ XA switch structure

Each resource manager participating in an externally coordinated unit of work must provide an XA switch structure. This structure defines both the capabilities of the resource manager and the functions that are to be called by the sync point coordinator.

IBM MQ provides two versions of this structure:

- *MQRMIXASwitch* for static XA resource management
- *MQRMIXASwitchDynamic* for dynamic XA resource management

Consult your transaction manager documentation to determine whether to use the static or dynamic resource management interface. Wherever a transaction manager supports it, we recommend that you use dynamic XA resource management.

Some 64-bit transaction managers treat the *long* type in the XA specification as 64-bit, and some treat it as 32-bit. IBM MQ supports both models:

- If your transaction manager is 32-bit, or your transaction manager is 64-bit but treats the *long* type as 32-bit, use the switch load file listed in [Table 5 on page 77](#).
- If your transaction manager is 64-bit and treats the *long* type as 64-bit, use the switch load file listed in [Table 6 on page 78](#).

Some 64-bit transaction managers treat the *long* type as 64-bit. The following 64-bit transaction managers are known to require the alternative 64-bit switch load file:

- Tuxedo
-   64 bit TXSeries

Consult your transaction manager documentation if you are unsure which model your transaction manager uses.










Table 5. XA switch load file names		
Platform	Switch load file name (server)	Switch load file name (extended transactional client)
 Windows	<i>mqmxa.dll</i>	<i>mqcxa.dll</i>
 AIX (nonthreaded)	<i>libmqmxa.a</i>	<i>libmqcxa.a</i>
 AIX (threaded)	<i>libmqmxa_r.a</i>	<i>libmqcxa_r.a</i>
 Linux (nonthreaded)	<i>libmqmxa.so</i>	<i>libmqcxa.so</i>
 Linux (threaded)	<i>libmqmxa_r.so</i>	<i>libmqcxa_r.so</i>

Table 6. Alternative 64-bit XA switch load file names

Platform	Switch load file name (server)	Switch load file name (extended transactional client)
 AIX (nonthreaded)	<i>libmqmxa64.a</i>	<i>libmqcxa64.a</i>
 AIX (threaded)	<i>libmqmxa64_r.a</i>	<i>libmqcxa64_r.a</i>
 Linux (nonthreaded)	<i>libmqmxa64.so</i>	<i>libmqcxa64.so</i>
 Linux (threaded)	<i>libmqmxa64_r.so</i>	<i>libmqcxa64_r.so</i>

Some external sync point coordinators (not CICS) require that each resource manager participating in a unit of work supplies its name in the name field of the XA switch structure. The IBM MQ resource manager name is MQSeries_XA_RMI.

The sync point coordinator defines how the IBM MQ XA switch structure links to it. Information about linking the IBM MQ XA switch structure with CICS is provided in [“Using CICS” on page 79](#). For information about linking the IBM MQ XA switch structure with other XA-compliant sync point coordinators, consult the documentation supplied with those products.

The following considerations apply to using IBM MQ with all XA-compliant sync point coordinators:

- It is expected that the transaction manager library code (running as part of their API that the application programmer has called) will call **xa_open** into IBM MQ at some point, before calling MQCONN.

The **xa_open** call must be made on the same thread where the MQCONN call is made. The reason for this requirement, is that the XA specification requires that the thread is used to imply context.

Note that this is the approach taken in sample program amqstxsx . c. This sample program assumes that an **xa_open** call is made into IBM MQ, from the library code of the transaction manager, within their function tpopen).

If no **xa_open** call is made on the same thread, before the MQCONN call, then the IBM MQ queue manager connection will not be associated with an XA context.

For more information, see [MQCTL](#).

- The xa_info structure passed on any xa_open call by the sync point coordinator includes the name of an IBM MQ queue manager. The name takes the same form as the queue manager name passed to the MQCONN call. If the name passed on the xa_open call is blank, the default queue manager is used.

Alternatively, the xa_info structure can contain values for the *TPM* and *AXLIB* parameters. The *TPM* parameter specifies the transaction manager being used. The valid values are CICS, TUXEDO and ENCINA. The *AXLIB* parameter specifies the name of the library that contains the transaction manager's ax_reg and ax_unreg functions. For more information on these parameters, see [Configuring an extended transactional client](#). If the xa_info structure contains either of these parameters, the queue manager name is specified in the QMNAME parameter, unless the default queue manager is being used.

- Only one queue manager at a time can participate in a transaction coordinated by an instance of an external sync point coordinator. The sync point coordinator is effectively connected to the queue manager, and is subject to the rule that only one connection at a time is supported.
- All applications that include calls to an external sync point coordinator can connect only to the queue manager that is participating in the transaction managed by the external coordinator (because they are already effectively connected to that queue manager). However, such applications must issue an MQCONN call to obtain a connection handle, and an MQDISC call before they exit.

- A queue manager with resource updates coordinated by an external sync point coordinator must start before the external sync point coordinator. Similarly, the sync point coordinator must end before the queue manager.
- If your external sync point coordinator terminates abnormally, stop and restart your queue manager before restarting the sync point coordinator to ensure that any messaging operations uncommitted at the time of the failure are properly resolved.

Using CICS

CICS is one of the elements of TXSeries.

The versions of TXSeries that are XA-compliant (and use a two-phase commit process) are defined in the [System Requirements for IBM MQ](#) information.

IBM MQ also supports other transaction managers. See the [System Requirements for IBM MQ](#) web page for links to information about supported software.

Requirements of the two-phase commit process

Requirements of the two-phase commit process when you use the CICS two-phase commit process with IBM MQ. These requirements do not apply to z/OS®.


Note the following requirements:

- IBM MQ and CICS must reside on the same physical machine.
- IBM MQ does not support CICS on an IBM MQ MQI client.
- You must start the queue manager, with its name specified in the XAD resource definition stanza, **before** you attempt to start CICS. Failure to do this will prevent you from starting CICS if you have added an XAD resource definition stanza for IBM MQ to the CICS region.
- Only one IBM MQ queue manager can be accessed at a time from a single CICS region.
- A CICS transaction must issue an MQCONN request before it can access IBM MQ resources. The MQCONN call must specify the name of the IBM MQ queue manager specified on the XAOpen entry of the XAD resource definition stanza for the CICS region. If this entry is blank, the MQCONN request must specify the default queue manager.
- A CICS transaction that accesses IBM MQ resources must issue an MQDISC call from the transaction before returning to CICS. Failure to do this might mean that the CICS application server is still connected, leaving queues open. Additionally, if you do not install a task termination exit (see [“Sample task termination exit” on page 82](#)), the CICS application server might later end abnormally, perhaps during a subsequent transaction.
- You must ensure that the CICS user ID (cics) is a member of the mqm group, so that the CICS code has the authority to call IBM MQ.



For transactions running in a CICS environment, the queue manager adapts its methods of authorization and determining context as follows:

- The queue manager queries the user ID under which CICS runs the transaction. This is the user ID checked by the Object Authority Manager, and is used for context information.
- In the message context, the application type is MQAT_CICS.
- The application name in the context is copied from the CICS transaction name.

General XA support

An XA switch load module is provided to enable you to link CICS with IBM MQ for AIX, Linux, and Windows systems. Additionally, sample source code files are provided to enable you to develop the XA switches for other transaction messages.  General XA is not supported on IBM i.




The names of the switch load modules provided are as follows:

Table 7. Essential code for CICS applications: XA initialization routine	
C (source)	C (exec) - add one of the following to your XAD.Stanza
amqzscix.c	 amqzsc - TXSeries for AIX 5.1
amqzscin.c	 mqmc4swi - TXSeries for Windows 5.1

Building libraries for use with TXSeries for Multiplatforms

Use this information when building libraries for use with TXSeries for Multiplatforms.

Pre-built switch load files are shared libraries (called *DLLs* on the Windows system) that you can use with CICS programs, which require a 2-phase commit transaction by using the XA protocol. The names of these pre-built libraries are in the table [Essential code for CICS applications: XA initialization routine](#). Sample source code is also supplied in the following directories:

Table 8. Installation directories on AIX, Linux, and Windows operating systems		
Platform	Directory	Source file
  AIX and Linux	MQ_INSTALLATION_PATH/samp/	amqzscix.c
 Windows	MQ_INSTALLATION_PATH\Tools\ c\Samples	amqzscin.c

where *MQ_INSTALLATION_PATH* is the directory in which you installed IBM MQ.

Note: You must use a CICS switch load file that has been built against the version of IBM MQ that the switch load file is used with.

To build the switch load file from the sample source, follow the instructions appropriate for your operating system:

AIX

Issue the following command:

```
export MQM_HOME=/usr/mqm
echo "amqzscix" > tmp.exp
xlc_r $MQM_HOME/samp/amqzscix.c -I/usr/lpp/cics/include -I$MQM_HOME/inc -e amqzscix -bE:tmp.exp -bM:SRE
-o amqzsc /usr/lpp/cics/lib/regxa_swxa.o -L$MQM_HOME/lib -L/usr/lpp/cics/lib -lcicsrt -lEncina
-lEncServer -lpthreads -lsarpc -lmqmcics_r -lmqmxr -lmqzi_r -lmqmcs_r
rm tmp.exp
```

Linux platforms

Issue the following command:

```
gcc -m32 -shared -fPIC -o amqzscix amqzscix.c
\ -IMQ_INSTALLATION_PATH/inc -I CICS_INSTALLATION_PATH/include
\ -LMQ_INSTALLATION_PATH/lib -Wl,-rpath=MQ_INSTALLATION_PATH/lib
\ -Wl,-rpath=/usr/lib -Wl,-rpath-link,/usr/lib -Wl,--no-undefined
-Wl,--allow-shlib-undefined \-L CICS_LIB_PATH/regxa_swxa.o \-lpthread -ldl -lc
-shared -lmqzi_r -lmqmxr -lmqmcics_r -ldl -lc
```

V9.4.0 Linux 64 bit TXSeries

Issue the following command:

```
gcc -shared -fPIC -o amqzscix amqzscix.c
\ -IMQ_INSTALLATION_PATH/inc -I CICS_INSTALLATION_PATH/include
\ -LMQ_INSTALLATION_PATH/lib64 -Wl,-rpath=MQ_INSTALLATION_PATH/lib64
```



```
\-Wl,-rpath=/usr/lib64 -Wl,-rpath-link,/usr/lib64 -Wl,--no-undefined -Wl,--allow-shlib-undefined
\ -L CICS_LIB_PATH/regxa_swx.o \-lpthread -ldl -lc
-shared -lmqzi_r -lmqmx64_r -lmqmcics_r -ldl -lc
```

Windows Windows

Follow these steps:

1. Use the `cl` command to build `amqzscin.obj` by compiling at least the following variables:

```
cl.exe -c -I EncinaPath\include -I MQ_INSTALLATION_PATH\include -Gz -LD amqzscin.c
```

2. Create a module definition file named `mqmc1415.def`, which contains the following lines:

```
LIBRARY MQMC4SWI
EXPORTS
CICS_XA_Init
```

3. Use the **lib** command to build an export file and an import library by using at least the following option:

```
lib -def:mqmc4swi.def -out:mqmc4swi.lib
```

If the `lib` command is successful then an `mqmc4swi.exp` file is also built.

4. Use the `link` command to build `mqmc4swi.dll` by using at least the following option:

```
link.exe -dll -nod -out:mqmc4swi.dll
amqzscin.obj CicsPath\lib\regxa_swx.obj
mqmc4swi.exp mqmcics4.lib
CicsPath\lib\libcicsrt.lib
DcePath\lib\libdce.lib DcePath\lib\pthreads.lib
EncinaPath\lib\libEncina.lib
EncinaPath\lib\libEncServer.lib
msvcrt.lib kernel32.lib
```

IBM MQ XA support and Tuxedo

IBM MQ on AIX, Linux, and Windows systems can block Tuxedo-coordinated XA applications indefinitely in `xa_start`.

This can occur only when two or more processes coordinated by Tuxedo in a single global transaction attempt to access IBM MQ using the same transaction branch ID (XID). If Tuxedo gives each process in the global transaction a different XID to use with IBM MQ, this cannot occur.

To avoid the problem, configure each application in Tuxedo that accesses IBM MQ under a single global transaction ID (gtrid), within its own Tuxedo server group. Processes in the same server group use the same XID when accessing resource managers on behalf of a single gtrid, and are therefore vulnerable to blocking in `xa_start` in IBM MQ. Processes in different server groups use separate XIDs when accessing resource managers and so do not have to serialize their transaction work in IBM MQ.

Enabling the CICS two-phase commit process

To enable CICS to use a two-phase commit process to coordinate transactions that include MQI calls, add a CICS XAD resource definition stanza entry to the CICS region. Note, this topic is not applicable to z/OS.

Here is an example of adding an XAD stanza entry for IBM MQ for Windows, where *Drive* is the drive where IBM MQ is installed (for example, D:).

```
cicsadd -cxad -rcics_region \
ResourceDescription="MQM XA Product Description" \
SwitchLoadFile="Drive:\Program Files\IBM\IBM MQ\bin\mqmc4swi.dll" \
XAOpen=queue_manager_name
```

For extended transactional clients, use the switch load file `mqcc4swi.dll`.

Here is an example of adding an XAD stanza entry for IBM MQ for AIX or Linux systems, where `MQ_INSTALLATION_PATH` represents the high-level directory in which IBM MQ is installed:

```
cicsadd -cxad -rcics_region \
  ResourceDescription="MQM XA Product Description" \
  SwitchLoadFile="MQ_INSTALLATION_PATH/lib/amqzsc" \
  XAOpen=queue_manager_name
```

For extended transactional clients, use the switch load file `amqczsc`.

See the CICS documentation for information about using the [cicsadd](#) command.

Calls to IBM MQ can be included in a CICS transaction, and the IBM MQ resources will be committed or rolled back as directed by CICS. This support is not available to client applications.

You must issue an MQCONN from your CICS transaction in order to access IBM MQ resources, followed by a corresponding MQDISC on exit.

Enabling CICS user exits

A CICS user exit *point* (normally referred to as a *user exit*) is a place in a CICS module at which CICS can transfer control to a program that you have written (a *user exit program*), and at which CICS can resume control when your exit program has finished its work.

Before using a CICS user exit, read the *CICS Administration Guide* for your platform.

Sample task termination exit

IBM MQ supplies sample source code for a CICS task termination exit.

The sample source code is in the following directories:

Table 9. CICS task termination exits		
Platform	Directory	Source file
AIX and Linux systems	<code>MQ_INSTALLATION_PATH/samp</code>	<code>amqzscgx.c</code>
Windows	<code>MQ_INSTALLATION_PATH\Tools\c\Samples</code>	<code>amqzscgn.c</code>

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

The build instructions for the sample task termination exit are contained in the comments in each source file.

This exit is invoked by CICS at normal and abnormal task termination (after any sync point has been taken). No recoverable work is permitted in the exit program.

These functions are only used in an IBM MQ and CICS context in which the CICS version supports the XA interface. CICS refers to these libraries as `programs` or `user exits`.

CICS has a number of user exits and `amqzscgx`, if used, is defined and enabled on CICS as the Task termination user exit (UE014015), that is, exit number 15.

When the task termination exit is called by CICS, CICS has already informed IBM MQ of the task's termination state and IBM MQ has taken the appropriate action (commit or rollback). All the exit does is to issue an MQDISC to clean up.

One purpose of installing and configuring your CICS system to use a task termination exit is to protect your system against some of the consequences of faulty application code. For example, if your CICS transaction ends abnormally without first calling MQDISC, and has no task termination exit installed, then you might see (within around 10 seconds) a subsequent unrecoverable failure of the CICS region. This is because IBM MQ health thread, which runs in the `cicsas` process, will not have been posted and given time to clean up and return. The symptoms might be that the `cicsas` process ends immediately, having written FFST reports to `/var/mqm/errors` or the equivalent location on Windows.

Using the Microsoft Transaction Server (COM+)

COM+ (Microsoft Transaction Server) is designed to help users run business logic applications in a typical middle tier server.

See [Features that can be used only with the primary installation on Windows](#) for important information.

COM+ divides work up into *activities*, which are typically short independent chunks of business logic, such as *transfer funds from account A to account B*. COM+ relies heavily on object orientation and in particular on COM; loosely a COM+ activity is represented by a COM (business) object.

COM+ is an integrated part of the operating system.

COM+ provides three services to the business object administrator, removing much of the worry from the business object programmer:

- Transaction management
- Security
- Resource pooling

You usually use COM+ with front-end code that is a COM client to the objects held within COM+, and back-end services such as a database, with IBM MQ bridging between the COM+ business object and the back-end.

The front-end code can be a stand-alone program, or an Active Server Page (ASP) hosted by the Microsoft Internet Information Server (IIS). The frontend code can be on the same computer as COM+ and its business objects, with connection through COM. Alternatively, the frontend code can be on a different computer, with connection through DCOM. You can use different clients to access the same COM+ business object in different situations.

The backend code can be on the same computer as COM+ and its business objects, or on a different computer with connection through any of the IBM MQ supported protocols.

Expiring global units of work

The queue manager can be configured to expire global units of work after an interval of inactivity.

You should enable an expiry scan, if your external Transaction Manager software has problems where it ends without first completing all transactions begun by its applications.

To enable expiry scanning, set the following variables in the environment from which your queue manager starts:

- **AMQ_TRANSACTION_EXPIRY_RESCAN**=*rescan interval in milliseconds*
- **AMQ_XA_TRANSACTION_EXPIRY**=*timeout interval in milliseconds*

Example for Linux and AIX:

```
# Scan runs every 30 seconds.
# Value is given in milliseconds.
export AMQ_TRANSACTION_EXPIRY_RESCAN=30000

# Transactions of age 60 seconds in the Idle state are regarded as expired.
# Value is given in milliseconds.
export AMQ_XA_TRANSACTION_EXPIRY=60000
```

Example for Windows:

```
rem Scan runs every 30 seconds.
rem Value is given in milliseconds.
set AMQ_TRANSACTION_EXPIRY_RESCAN=30000

rem Transactions of age 60 seconds in the Idle state are regarded as expired.
rem Value is given in milliseconds.
set AMQ_XA_TRANSACTION_EXPIRY=60000
```

Note: The expiry scan only affects transactions begun by external Transaction Manager software that are in Idle state in Table 6-4 of the [XA Specification](#) from [The OPEN Group](#).

Transactions in Idle state are not associated on any application thread, and the external Transaction Manager software has not yet called the **xa_prepare** function call to place the transaction in Prepared state.

If the external Transaction Manager software crashes and restarts, then normally it does not deliver a Commit or Rollback for transactions in this state.

The queue manager will roll back Puts and Gets performed within a transaction that has reached Idle state, if you:

- Restart the queue manager.
- Enable the expiry scan, and an Idle transaction is found that is older than the specified age.

Set the **AMQ_XA_TRANSACTION_EXPIRY** to allow for the expected interval between an application performing its work within the transaction, and completing the transaction.

Set the **AMQ_TRANSACTION_EXPIRY_RESCAN** value to a lower value than the **AMQ_XA_TRANSACTION_EXPIRY** value. Do this so that the scans of the transaction lists occur more than once within the **AMQ_XA_TRANSACTION_EXPIRY** interval.

How to decide if you need to enable expiry scans

You only need expiry scans if you have an external Transaction Manager, and it is ended or crashed without completing its Idle transactions.

If the external Transaction Manager ends in that way, your application's Puts and Gets of messages are not committed or rolled back, unless you restart the queue manager or have configured expiry scans.

If there are transactions in this state, then the output of **dspmqtzn -m QMNAME -q -a** includes output like the example below. There are a few things to look out for, to be sure this is a transaction of concern:

- The TRANSTATE is ACTIVE, not PREPARED.
- The timestamps show the transaction was started a long time before you ran **dspmqtzn**. If the timestamps are very recent, then this is not a transaction of concern.
- The EXTURID shows this is an XA transaction because it has non-empty values for XA_FORMATID, XA_GTRID and XA_BQUAL.
- There are one or more application queue names listed as OBJECT records, showing that Puts or Gets have been done on these queues, under this transaction.
- There is no live-connection information in this record (for example: CONN, PID, TID, APPLTAG, USERID) because the application is no longer connected.

```
TranNum(0,513)
TRANSTATE(ACTIVE)
UOWLOGDA(2024-05-23) UOWLOGTI(17.03.22)
UOWSTDA(2024-05-23) UOWSTTI(17.03.22)
UOWLOG( )
EXTURID(XA_FORMATID[WASD] XA_GTRID[0000018] XA_BQUAL[0000018])
OBJECT(MY.QUEUE)
```

Unit of recovery disposition

IBM MQ for z/OS provides unit of recovery dispositions. This feature allows you to configure whether the second phase of 2-phase commit transactions can be driven, for example, during recovery, when connected to another queue manager within the same queue sharing group (QSG).

IBM MQ for z/OS V7.0.1 and later supports the unit of recovery disposition.

Unit of recovery disposition

The unit of recovery disposition is related to an application's connection and subsequently any transactions that it starts. There are two possible unit of recovery dispositions.

- A GROUP unit of recovery disposition identifies that a transactional application is logically connected to the queue sharing group and does not have an affinity to any specific queue manager. Any 2-phase commit transactions that it starts that have completed phase-1 of the commit process, that is, they are in doubt, can be inquired and resolved, when connected to any queue manager within the QSG. In a recovery scenario this means that the transaction coordinator does not have to reconnect to the same queue manager, which might be unavailable.
- A QMGR unit of recovery disposition identifies that an application has a direct affinity to the queue manager to which it is connected and any transactions that it starts also have this disposition.

In a recovery scenario the transaction coordinator must reconnect to the same queue manager to inquire, and resolve, any in-doubt transactions, irrespective of whether the queue manager belongs to a queue sharing group.

z/OS For details of how to implement this feature, see [Unit of recovery disposition in a queue sharing group](#).

Security scenarios

A set of scenarios that demonstrate applying security to different configurations.

The available security scenarios are described in the following subtopics:

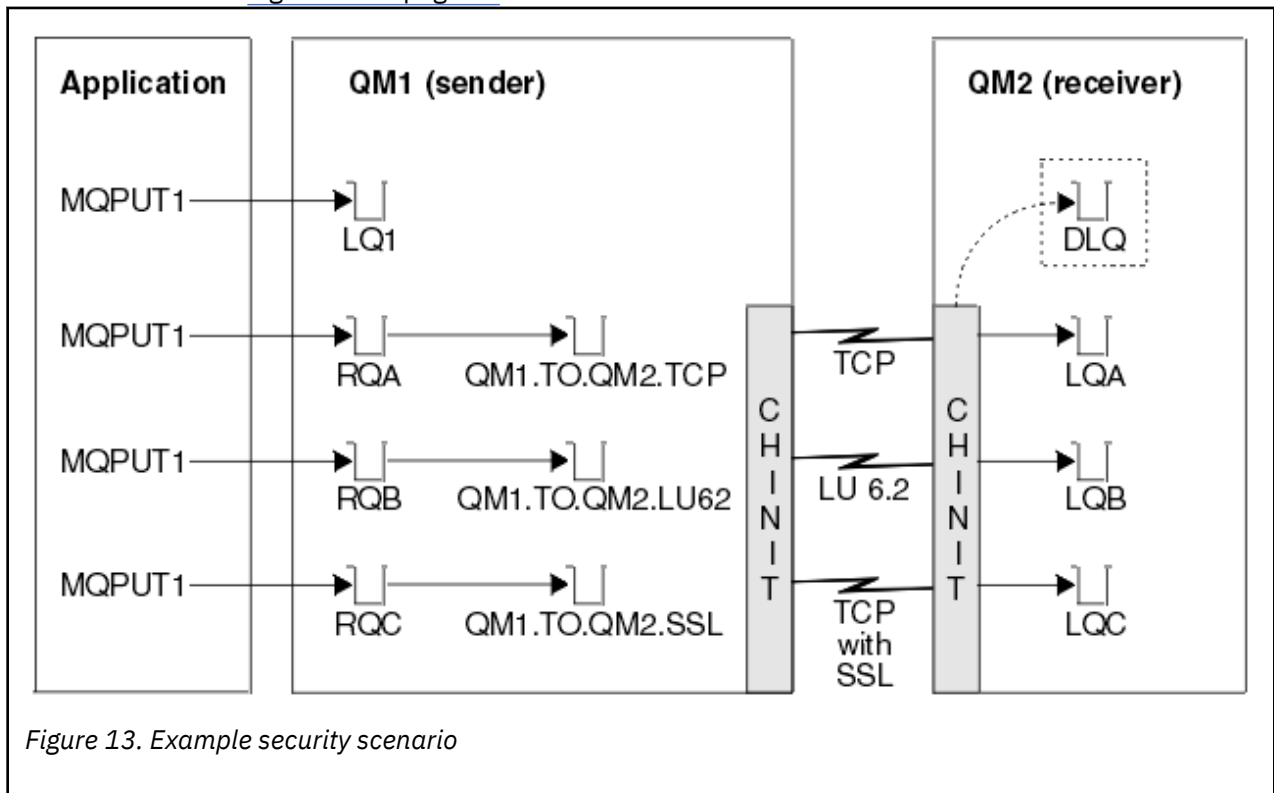
Related tasks

z/OS [Setting up security on z/OS](#)

z/OS Security scenario: two queue managers on z/OS

In this scenario, an application uses the **MQPUT1** call to put messages to queues on queue manager QM1. Some of the messages are then forwarded to queues on QM2, using TCP and LU 6.2 channels. The TCP channels can either use SSL/TLS or not. The application could be a batch application or a CICS application, and the messages are put using the MQPMO_SET_ALL_CONTEXT option.

This is illustrated in Figure 13 on page 85.



The following assumptions are made about the queue managers:

- All the required IBM MQ definitions have been predefined or have been made through the CSQINP2 data set processed at queue manager startup.


If they have not, you need the appropriate access authority to the commands needed to define these objects.

- All the RACF® profiles required have been defined and appropriate access authorities have been granted, before the queue manager and channel initiators started.

If they have not, you need the appropriate authority to issue the RACF commands required to define all the profiles needed and grant the appropriate access authorities to those profiles. You also need the appropriate authority to issue the MQSC security commands to start using the new security profiles.

- All digital certificates required have been created and connected to key rings. The digital certificate sent by QM1 as part of the SSL/TLS handshake is recognized by RACF on QM2's system, either because it is also installed in that RACF profile, or because a matching Certificate Name File (CNF) filter exists.

Related tasks

 [Setting up security on z/OS](#)

Security switch settings for two queue manager scenario

Switch settings and RACF profiles.

The following security switches are set for both queue managers:

- Subsystem security on
- Queue security on
- Alternate user security on
- Context security on
- Process security off
- Namelist security off
- Topic security off
- Connection security on
- Command security on
- Command resource security on

The following profiles are defined in the MQADMIN class to turn off process, namelist and topic security:

```
QM1.NO.PROCESS.CHECKS
QM1.NO.NLIST.CHECKS
QM1.NO.TOPIC.CHECKS
QM2.NO.PROCESS.CHECKS
QM2.NO.NLIST.CHECKS
QM2.NO.TOPIC.CHECKS
```

Queue manager QM1 in two queue manager scenario

Queues and channels for QM1.

The following queues are defined on queue manager QM1:

LQ1

A local queue.

RQA

A remote queue definition, with the following attributes:

- RNAME(LQA)

- RQMNAME(QM2)
- XMITQ(QM1.TO.QM2.TCP)

RQB

A remote queue definition, with the following attributes:

- RNAME(LQB)
- RQMNAME(QM2)
- XMITQ(QM1.TO.QM2.LU62)

RQC

A remote queue definition, with the following attributes:

- RNAME(LQC)
- RQMNAME(QM2)
- XMITQ(QM1.TO.QM2.TLS)

QM1.TO.QM2.TCP

A transmission queue.

QM1.TO.QM2.LU62

A transmission queue.

QM1.TO.QM2.TLS

A transmission queue.

The following channels are defined on QM1:

QM1.TO.QM2.TCP

A sender channel definition, with the following attributes:

- CHLTYPE(SDR)
- TRPTYPE(TCP)
- XMITQ(QM1.TO.QM2.TCP)
- CONNAME(QM2TCP)

QM1.TO.QM2.LU62

A sender channel definition, with the following attributes:

- CHLTYPE(SDR)
- TRPTYPE(LU62)
- XMITQ(QM1.TO.QM2.LU62)
- CONNAME(QM2LU62)

(See [Security considerations for the channel initiator on z/OS](#) for information about setting up APPC security.)

QM1.TO.QM2.TLS

A sender channel definition, with the following attributes:

- CHLTYPE(SDR)
- TRPTYPE(TCP)
- XMITQ(QM1.TO.QM2.TLS)
- CONNAME(QM2TCP)
- SSLCIPH(TLS_RSA_WITH_AES_128_CBC_SHA256)

Queue manager QM2 in two queue manager scenario

Queues and channels for QM2.

The following queues have been defined on queue manager QM2:

LQA

A local queue.

LQB

A local queue.

LQC

A local queue.

DLQ

A local queue that is used as the dead-letter queue.

The following channels have been defined on QM2:

QM1.TO.QM2.TCP

A receiver channel definition, with the following attributes:

- CHLTYPE(RCVR)
- TRPTYPE(TCP)
- PUTAUT(CTX)
- MCAUSER(MCATCP)

QM1.TO.QM2.LU62

A receiver channel definition, with the following attributes:

- CHLTYPE(RCVR)
- TRPTYPE(LU62)
- PUTAUT(CTX)
- MCAUSER(MCALU62)

(See [Security considerations for the channel initiator on z/OS](#) for information about setting up APPC security.)

QM1.TO.QM2.TLS

A receiver channel definition, with the following attributes:

- CHLTYPE(RCVR)
- TRPTYPE(TCP)
- PUTAUT(CTX)
- MCAUSER(MCASSL)
- SSLCIPH(TLS_RSA_WITH_AES_128_CBC_SHA256)

User IDs used in two queue manager scenario

Explanation of the user IDs in the scenario.

The following user IDs are used:

BATCHID

Batch application (Job or TSO ID)

MSGUSR

UserIdentifier in MQMD (context user ID)

MOVER1

QM1 channel initiator address space user ID

MOVER2

QM2 channel initiator address space user ID

MCATCP

MCAUSER specified on the TCP/IP without SSL/TLS receiver channel definition

MCAU62

MCAUSER specified on the LU 6.2 receiver channel definition

MCASSL

MCAUSER specified on the TCP/IP with SSL/TLS receiver channel definition

CICSAD1

CICS address space ID

CICSTX1

CICS task user ID

CERTID

The user ID associated by RACF with the flowed certificate.

Security profiles and accesses required for the two queue manager scenario

Information about the security profiles and accesses required for either a batch or CICS implementation of the two queue manager scenario.

The following table shows the security profiles that are required to enable the two queue manager scenario to work. Additional security profiles are also needed, depending on whether you are doing a batch or a CICS implementation of the scenario. For further information see [“Security profiles required for a batch application” on page 90](#) and [“Security profiles required for a CICS application” on page 92](#).

Table 10. Security profiles for the example scenario.			
The four columns in this table show the class, the profile, the user ID, and access for the two queue manager scenario.			
Class	Profile	User ID	Access
MQCONN	QM1.CHIN	MOVER1	READ
MQADMIN	QM1.RESLEVEL	BATCHID CICSAD1 MOVER1	NONE
MQADMIN	QM1.CONTEXT.**	MOVER1	CONTROL
MQQUEUE	QM1.SYSTEM.COMMAND.INPUT	MOVER1	UPDATE
MQQUEUE	QM1.SYSTEM.CHANNEL.SYNCQ	MOVER1	UPDATE
MQQUEUE	QM1.SYSTEM.CHANNEL.INITQ	MOVER1	UPDATE
MQQUEUE	QM1.SYSTEM.COMMAND.REPLY.MODEL	MOVER1	UPDATE
MQQUEUE	QM1.SYSTEM.ADMIN.CHANNEL.EVENT	MOVER1	UPDATE
MQQUEUE	QM1.QM1.TO.QM2.TCP	MOVER1	ALTER
MQQUEUE	QM1.QM1.TO.QM2.LU62	MOVER1	ALTER
MQQUEUE	QM1.QM1.TO.QM2.TLS	MOVER1	ALTER
MQCONN	QM2.CHIN	MOVER2	READ
MQADMIN	QM2.RESLEVEL	MOVER2	NONE
MQADMIN	QM2.CONTEXT.**	MOVER2	CONTROL
MQQUEUE	QM2.SYSTEM.COMMAND.INPUT	MOVER2	UPDATE
MQQUEUE	QM2.SYSTEM.CHANNEL.SYNCQ	MOVER2	UPDATE
MQQUEUE	QM2.SYSTEM.CHANNEL.INITQ	MOVER2	UPDATE
MQQUEUE	QM2.SYSTEM.COMMAND.REPLY.MODEL	MOVER2	UPDATE

Table 10. Security profiles for the example scenario.

The four columns in this table show the class, the profile, the user ID, and access for the two queue manager scenario.

(continued)

Class	Profile	User ID	Access
MQQUEUE	QM2.SYSTEM.ADMIN.CHANNEL.EVENT	MOVER2	UPDATE
MQQUEUE	QM2.DLQ	MOVER2	UPDATE

Security profiles required for a batch application

Additional security profiles required for a batch implementation of the two queue manager scenario.

The batch application runs under user ID BATCHID on QM1. It connects to queue manager QM1 and puts messages to the following queues:

- LQ1
- RQA
- RQB
- RQC

It uses the MQPMO_SET_ALL_CONTEXT option. The alternate user ID found in the *UserIdentifier* field of the message descriptor (MQMD) is MSGUSR.

The following profiles are required on queue manager QM1:

Table 11. Sample security profiles for the batch application on queue manager QM1

Class	Profile	User ID	Access
MQCONN	QM1.BATCH	BATCHID	READ
MQADMIN	QM1.CONTEXT.**	BATCHID	CONTROL
MQQUEUE	QM1.LQ1	BATCHID	UPDATE
MQQUEUE	QM1.RQA	BATCHID	UPDATE
MQQUEUE	QM1.RQB	BATCHID	UPDATE
MQQUEUE	QM1.RQC	BATCHID	UPDATE

The following profiles are required on queue manager QM2 for messages put to queue RQA on queue manager QM1 (for the TCP/IP channel not using TLS):

Table 12. Sample security profiles for queue manager QM2 using TCP/IP and not TLS

Class	Profile	User ID	Access
MQADMIN	QM2.ALTERNATE.USER.MSGUSR	MCATCP MOVER2	UPDATE
MQADMIN	QM2.CONTEXT.**	MCATCP MOVER2	CONTROL
MQQUEUE	QM2.LQA	MOVER2 MSGUSR	UPDATE
MQQUEUE	QM2.DLQ	MOVER2 MSGUSR	UPDATE

Notes:

1. The user ID passed in the MQMD of the message is used as the user ID for the MQPUT1 on queue manager QM2 because the receiver channel was defined with PUTAUT(CTX) and MCAUSER(MCATCP).

2. The MCAUSER field of the receiver channel definition is set to MCATCP; this user ID is used in addition to the channel initiator address space user ID for the checks carried out against the alternate user ID and context profile.
3. The MOVER2 user ID and the *UserIdentifier* in the message descriptor (MQMD) are used for the resource checks against the queue.
4. The MOVER2 and MSGUSR user IDs both need access to the dead-letter queue so that messages that cannot be put to the destination queue can be sent there.
5. Two user IDs are checked on all three checks performed because RESLEVEL is set to NONE.

The following profiles are required on queue manager QM2 for messages put to queue RQB on queue manager QM1 (for the LU 6.2 channel):

Table 13. Sample security profiles for queue manager QM2 using LU 6.2			
Class	Profile	User ID	Access
MQADMIN	QM2.ALTERNATE.USER.MSGUSR	MCALU62 MOVER1	UPDATE
MQADMIN	QM2.CONTEXT.**	MCALU62 MOVER1	CONTROL
MQQUEUE	QM2.LQB	MOVER1 MSGUSR	UPDATE
MQQUEUE	QM2.DLQ	MOVER1 MSGUSR	UPDATE

Notes:

1. The user ID passed in the MQMD of the message is used as the user ID for the MQPUT1 on queue manager QM2 because the receiver channel was defined with PUTAUT(CTX) and MCAUSER(MCALU62).
2. The MCA user ID is set to the value of the MCAUSER field of the receiver channel definition (MCALU62).
3. Because LU 6.2 supports security on the communications system for the channel, the user ID received from the network is used as the channel user ID (MOVER1).
4. Two user IDs are checked on all three checks performed because RESLEVEL is set to NONE.
5. MCALU62 and MOVER1 are used for the checks performed against the alternate user ID and Context profiles, and MSGUSR and MOVER1 are used for the checks against the queue profile.
6. The MOVER1 and MSGUSR user IDs both need access to the dead-letter queue so that messages that cannot be put to the destination queue can be sent there.

The following profiles are required on queue manager QM2 for messages put to queue RQC on queue manager QM1 (for the TCP/IP channel using TLS):

Table 14. Sample security profiles for queue manager QM2 using TCP/IP and TLS			
Class	Profile	User ID	Access
MQADMIN	QM2.ALTERNATE.USER.MSGUSR	MCASSL CERTID	UPDATE
MQADMIN	QM2.CONTEXT.**	MCASSL CERTID	CONTROL
MQQUEUE	QM2.LQC	CERTID MSGUSR	UPDATE
MQQUEUE	QM2.DLQ	CERTID MSGUSR	UPDATE

Notes:

1. The user ID passed in the MQMD of the message is used as the user ID for the MQPUT1 on queue manager QM2 because the receiver channel was defined with PUTAUT(CTX) and MCAUSER(MCASSL).

2. The MCA user ID is set to the value of the MCAUSER field of the receiver channel definition (MCASSL).
3. Because the certificate flowed by the channel from QM1 as part of the TLS handshake might be installed on QM2's system, or might match a certificate name filter on QM2's system, the user ID found during that matching is used as the channel user ID (CERTID).
4. Two user IDs are checked on all three checks performed because RESLEVEL is set to NONE.
5. MCASSL and CERTID are used for the checks performed against the alternate user ID and Context profiles, and MSGUSR and MOVER1 are used for the checks against the queue profile.
6. The CERTID and MSGUSR user IDs both need access to the dead-letter queue so that messages that cannot be put to the destination queue can be sent there.

Security profiles required for a CICS application

Additional security profiles required for a CICS implementation of the two queue manager scenario.

The CICS application uses a CICS address space user ID of CICSAD1 and a CICS task user ID of CICSTX1. The security profiles required on queue manager QM1 are different from those profiles required for the batch application. The profiles required on queue manager QM2 are the same as for the batch application.

The following profiles are required on queue manager QM1:

Table 15. Sample security profiles for the CICS application on queue manager QM1			
Class	Profile	User ID	Access
MQCONN	QM1.CICS	CICSAD1	READ
MQADMIN	QM1.CONTEXT.**	CICSAD1 CICSTX1	CONTROL
MQQUEUE	QM1.LQ1	CICSAD1 CICSTX1	UPDATE
MQQUEUE	QM1.RQA	CICSAD1 CICSTX1	UPDATE
MQQUEUE	QM1.RQB	CICSAD1 CICSTX1	UPDATE

Security scenario: queue sharing group on z/OS

In this scenario, an application uses the **MQPUT1** call to put messages to queues on queue manager QM1. Some of the messages are then forwarded to queues on QM2, using TCP and LU 6.2 channels. The application is a batch application, and the messages are put using the MQPMO_SET_ALL_CONTEXT option.

This is illustrated in [Figure 13 on page 85](#).

The following assumptions are made about the queue managers:


- All the required IBM MQ definitions have been predefined or have been made through the CSQINP2 data set processed at queue manager startup.

If they have not, you need the appropriate access authority to the commands needed to define these objects.

- All the RACF profiles required have been defined and appropriate access authorities have been granted, before the queue manager and channel initiators started.

If they have not, you need the appropriate authority to issue the RACF commands required to define all the profiles needed and grant the appropriate access authorities to those profiles. You also need the appropriate authority to issue the MQSC security commands to start using the new security profiles.

Related tasks

 [Setting up security on z/OS](#)

Security switch settings for queue sharing group scenario

Switch settings and RACF profiles.

The following security switches are set for the queue sharing group:

- Subsystem security on
- Queue sharing group security on
- Queue manager security off
- Queue security on
- Alternate user security on
- Context security on
- Process security off
- Namelist security off
- Topic security off
- Connection security on
- Command security on
- Command resource security on

The following profiles are defined in the MQADMIN class to turn process, namelist, topic and queue manager level security off:

```
QSGA.NO.PROCESS.CHECKS
QSGA.NO.NLIST.CHECKS
QSGA.NO.TOPIC.CHECKS
QSGA.NO.QMGR.CHECKS
```

Queue manager QM1 in queue sharing group scenario

Queues and channels for QM1.

The following queues are defined on queue manager QM1:

LQ1

A local queue.

RQA

A remote queue definition, with the following attributes:

- RNAME(LQA)
- RQMNAME(QM2)
- XMITQ(QM1.TO.QM2.TCP)

RQB

A remote queue definition, with the following attributes:

- RNAME(LQB)
- RQMNAME(QM2)
- XMITQ(QM1.TO.QM2.LU62)

QM1.TO.QM2.TCP

A transmission queue.

QM1.TO.QM2.LU62

A transmission queue.

The following channels are defined on QM1:

QM1.TO.QM2.TCP

A sender channel definition, with the following attributes:

- CHLTYPE(SDR)
- TRPTYPE(TCP)
- XMITQ(QM1.TO.QM2.TCP)
- CONNAME(QM2TCP)

QM1.TO.QM2.LU62

A sender channel definition, with the following attributes:

- CHLTYPE(SDR)
- TRPTYPE(LU62)
- XMITQ(QM1.TO.QM2.LU62)
- CONNAME(QM2LU62)

(See [Security considerations for the channel initiator on z/OS](#) for information about setting up APPC security.)

Queue manager QM2 in queue sharing group scenario

Queues and channels for QM2.

The following queues have been defined on queue manager QM2:

LQA

A local queue.

LQB

A local queue.

DLQ

A local queue that is used as the dead-letter queue.

The following channels have been defined on QM2:

QM1.TO.QM2.TCP

A receiver channel definition, with the following attributes:

- CHLTYPE(RCVR)
- TRPTYPE(TCP)
- PUTAUT(CTX)
- MCAUSER(MCATCP)

QM1.TO.QM2.LU62

A receiver channel definition, with the following attributes:

- CHLTYPE(RCVR)
- TRPTYPE(LU62)
- PUTAUT(CTX)
- MCAUSER(MCALU62)

(See [Security considerations for the channel initiator on z/OS](#) for information about setting up APPC security.)

User IDs used in queue sharing group scenario

Explanation of the user IDs in the scenario.

The following user IDs are used:

BATCHID

Batch application (Job or TSO ID)

MSGUSR*UserIdentifier* in MQMD (context user ID)**MOVER1**

QM1 channel initiator address space user ID

MOVER2

QM2 channel initiator address space user ID

MCATCP

MCAUSER specified on the TCP/IP receiver channel definition


MCALU62

MCAUSER specified on the LU 6.2 receiver channel definition

Security profiles and accesses required for queue sharing group scenario


Security profiles and accesses for either a batch or CICS implementation of the queue sharing group scenario.

The following table shows the security profiles that are required to enable the queue sharing group scenario to work. A batch implementation of this scenario also requires the additional security profiles that are described in [“Security profiles required for a batch application”](#) on page 96.

Table 16. Security profiles for the example scenario. 

The four columns in this table show the class, the profile, the user ID, and access for the queue sharing group scenario.

Class	Profile	User ID	Access
MQCONN	QSGA.CHIN	MOVER1 MOVER2	READ
MQADMIN	QSGA.RESLEVEL	BATCHID MOVER1 MOVER2	NONE
MQADMIN	QSGA.CONTEXT.**	MOVER1 MOVER2	CONTROL
MQQUEUE	QSGA.SYSTEM.COMMAND.INPUT	MOVER1 MOVER2	UPDATE
MQQUEUE	QSGA.SYSTEM.CHANNEL.SYNCQ	MOVER1 MOVER	UPDATE
MQQUEUE	QSGA.SYSTEM.CHANNEL.INITQ	MOVER1 MOVER2	UPDATE
MQQUEUE	QSGA.SYSTEM.COMMAND.REPLY.MODEL	MOVER1 MOVER2	UPDATE
MQQUEUE	QSGA.SYSTEM.ADMIN.CHANNEL.EVENT	MOVER1 MOVER2	UPDATE
MQQUEUE	QSGA.SYSTEM.QSG.CHANNEL.SYNCQ	MOVER1 MOVER2	UPDATE
MQQUEUE	QSGA.SYSTEM.QSG.TRANSMIT.QUEUE	MOVER1 MOVER2	UPDATE
MQQUEUE	QSGA.QM1.TO.QM2.TCP	MOVER1	ALTER

Table 16. Security profiles for the example scenario. 

The four columns in this table show the class, the profile, the user ID, and access for the queue sharing group scenario.

(continued)

Class	Profile	User ID	Access
MQQUEUE	QSGA.QM1.TO.QM2.LU62	MOVER1	ALTER
MQQUEUE	QSGA.DLQ	MOVER2	UPDATE

Security profiles required for a batch application

Additional security profiles required for a batch implementation of the queue sharing group scenario.

The batch application runs under user ID BATCHID on QM1. It connects to queue manager QM1 and puts messages to the following queues:

- LQ1
- RQA
- RQB

It uses the MQPMO_SET_ALL_CONTEXT option. The user ID found in the *UserIdentifier* field of the message descriptor (MQMD) is MSGUSR.

The following profiles are required on queue manager QM1:

Table 17. Sample security profiles for the batch application on queue manager QM1

Class	Profile	User ID	Access
MQCONN	QSGA.BATCH	BATCHID	READ
MQADMIN	QSGA.CONTEXT.**	BATCHID	CONTROL
MQQUEUE	QSGA.LQ1	BATCHID	UPDATE
MQQUEUE	QSGA.RQA	BATCHID	UPDATE
MQQUEUE	QSGA.RQB	BATCHID	UPDATE

The following profiles are required on queue manager QM2 for messages put to queue RQA on queue manager QM1 (for the TCP/IP channel):

Table 18. Sample security profiles for queue manager QM2 using TCP/IP

Class	Profile	User ID	Access
MQADMIN	QSGA.ALTERNATE.USER.MSGUSR	MCATCP MOVER2	UPDATE
MQADMIN	QSGA.CONTEXT.**	MCATCP MOVER2	CONTROL
MQQUEUE	QSGA.LQA	MOVER2 MSGUSR	UPDATE
MQQUEUE	QSGA.DLQ	MOVER2 MSGUSR	UPDATE

Notes:

1. The user ID passed in the MQMD of the message is used as the user ID for the MQPUT1 on queue manager QM2 because the receiver channel was defined with PUTAUT(CTX) and MCAUSER(MCATCP).

2. The MCAUSER field of the receiver channel definition is set to MCATCP; this user ID is used in addition to the channel initiator address space user ID for the checks carried out against the alternate user ID and context profile.
3. The MOVER2 user ID and the *UserIdentifier* in the message descriptor (MQMD) are used for the resource checks against the queue.
4. The MOVER2 and MSGUSR user IDs both need access to the dead-letter queue so that messages that cannot be put to the destination queue can be sent there.
5. Two user IDs are checked on all three checks performed because RESLEVEL is set to NONE.

The following profiles are required on queue manager QM2 for messages put to queue RQB on queue manager QM1 (for the LU 6.2 channel):

Table 19. Sample security profiles for queue manager QM2 using LU 6.2			
Class	Profile	User ID	Access
MQADMIN	QSGA.ALTERNATE.USER.MSGUSR	MCALU62 MOVER1	UPDATE
MQADMIN	QSGA.CONTEXT.**	MCALU62 MOVER1	CONTROL
MQQUEUE	QSGA.LQB	MOVER1 MSGUSR	UPDATE
MQQUEUE	QSGA.DLQ	MOVER1 MSGUSR	UPDATE

Notes:

1. The user ID passed in the MQMD of the message is used as the user ID for the MQPUT1 on queue manager QM2 because the receiver channel was defined with PUTAUT(CTX) and MCAUSER(MCALU62).
2. The MCA user ID is set to the value of the MCAUSER field of the receiver channel definition (MCALU62).
3. Because LU 6.2 supports security on the communications system for the channel, the user ID received from the network is used as the channel user ID (MOVER1).
4. Two user IDs are checked on all three checks performed because RESLEVEL is set to NONE.
5. MCALU62 and MOVER1 are used for the checks performed against the alternate user ID and Context profiles, and MSGUSR and MOVER1 are used for the checks against the queue profile.
6. The MOVER1 and MSGUSR user IDs both need access to the dead-letter queue so that messages that cannot be put to the destination queue can be sent there.

Server-to-server message channel interception example configurations

Server-to-server message channel interception requires configuration of channel definitions, as well as Advanced Message Security policies, to ensure that inbound and outbound messages can be correctly protected or unprotected. The configuration varies depending on whether the channel is inbound or outbound.

Inbound channel

The following example shows a typical configuration for an inbound channel of type receiver, and provides details of the AMS policy required to protect unprotected inbound messages:



Figure 14. Inbound configuration

The example shows:

- Queue manager QMA
- Channel TO.QMA
- Local queue DESTQ

Use the following code:

```
DEFINE CHANNEL(TO.QMA) CHLTYPE(RCVR) SSLCAUTH(REQUIRED) SSLCIPH(ANY_TLS12) TRPTYPE(TCP)
SPLPROT(ASPOLICY)

DEFINE QLOCAL(DESTQ) DESCR('AMS PROTECTED QUEUE')

setmqsp1 -m QMA -p DESTQ -e AES256 -x CN=TEST,O=ORG,C=US
```

Note: The policy described in the preceding text encrypts messages only; that is, AMS Confidentiality.

See [setmqsp1](#) and [the message security policy \(CSQOUTIL\)](#) for information on using **setmqsp1** on z/OS.

Outbound channel

The following example shows a typical configuration for an outbound channel of type sender. The example provides details of the AMS policies required to protect messages put to the remote queue, and to unprotect and send messages got from the transmission queue:

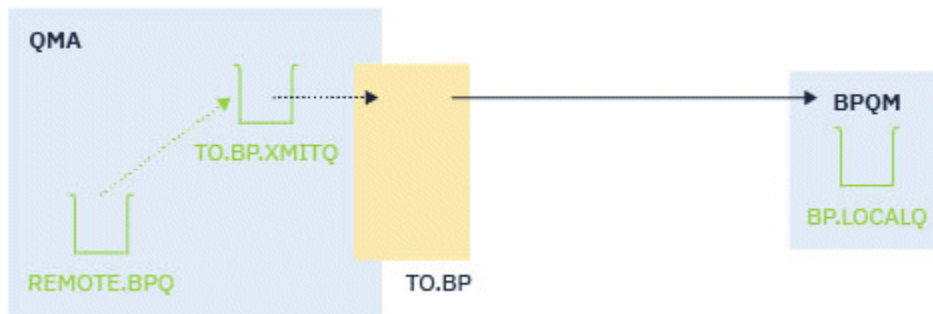


Figure 15. Outbound configuration

The example shows a:

- Queue manager QMA
- Channel TO.BP
- Local transmission queue TO.BP.XMITQ
- Remote queue REMOTE.BPQ

use the following code:

```
DEFINE CHANNEL(TO.BP) CHLTYPE(SDR) SSLCAUTH(REQUIRED) SSLCIPH(ANY_TLS12) TRPTYPE(TCP)
SPLPROT(REMOVE) CONNAME('server(1414)') XMITQ(TO.BP.XMITQ)

DEFINE QLOCAL(TO.BP.XMITQ) DESCR('TRANSMISSION QUEUE FOR TO.BP') USAGE(XMITQ)

DEFINE QREMOTE(REMOTE.BPQ) DESCR('REMOTE QUEUE TO BP') RNAME(BP.LOCALQ) RQMANME(BPQM)
XMITQ(TO.BP.XMITQ)
```

```
setmqsp1 -m QMA -p TO.BP.XMITQ -e AES256 -x CN=TEST,O=ORG,C=US
setmqsp1 -m QMA -p REMOTE.BPQ -e AES256 -x CN=TEST,O=ORG,C=US
```




Note: The policy described in the preceding text encrypts messages only; that is, AMS Confidentiality.

Connecting two queue managers using SSL/TLS

Secure communications that use the TLS cryptographic security protocols involve setting up the communication channels and managing the digital certificates that you will use for authentication.

To set up your SSL/TLS installation you must define your channels to use TLS. You must also obtain and manage your digital certificates. On a test system, you can use self-signed certificates or certificates issued by a local certificate authority (CA). On a production system, do not use self-signed certificates.


For full information about creating and managing certificates, see the following topics:

-  [Working with SSL or TLS on IBM i](#)
-  [Working with SSL or TLS on AIX, Linux, and Windows systems](#)
-  [Working with SSL or TLS on z/OS](#)

This collection of topics introduces the tasks involved in setting up SSL/TLS communications, and provides step-by-step guidance on completing those tasks.

You might also want to test SSL/TLS client authentication, which are an optional part of the protocols. During the SSL/TLS handshake, the SSL/TLS client always obtains and validates a digital certificate from the server. With the IBM MQ implementation, the SSL/TLS server always requests a certificate from the client.

Notes:

1. In this context, an SSL/TLS client refers to the connection initiating the handshake.
2.  When a z/OS queue manager is acting in the role of an SSL/TLS client, the queue manager sends only a certificate.

The SSL/TLS client sends a certificate only if it can find a certificate with a matching label. See [Digital certificate labels](#) for details.

The SSL/TLS server always validates the client certificate if one is sent. If the client does not send a certificate, authentication fails only if the end of the channel that is acting as the SSL/TLS server is defined with either the **SSLCAUTH** parameter set to REQUIRED or an **SSLPEER** parameter has a value set. For more information about connecting a queue manager anonymously, that is, when the SSL/TLS client does not send a certificate, see [“Connecting two queue managers using one-way authentication”](#) on page 104.

Using self-signed certificates for mutual authentication of two queue managers

Follow these sample instructions to implement mutual authentication between two queue managers, using self-signed TLS certificates.

About this task

Scenario:

- You have two queue managers, QM1 and QM2, which need to communicate securely. You require mutual authentication to be carried out between QM1 and QM2.
- You have decided to test your secure communication using self-signed certificates.

The resulting configuration looks like this:

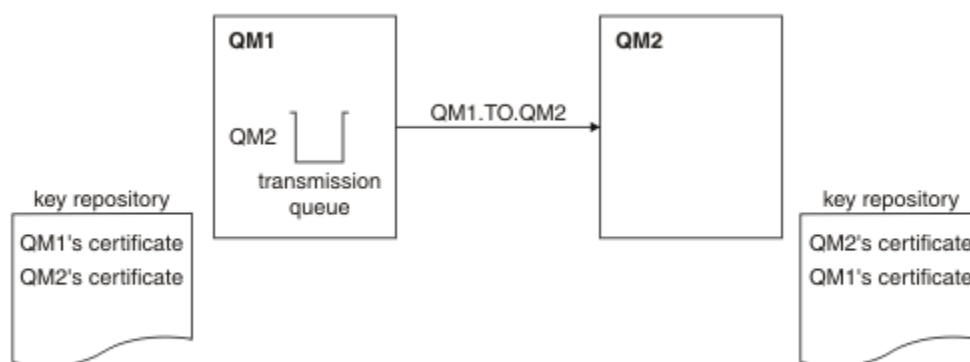


Figure 16. Configuration resulting from this task

In Figure 16 on page 100, the key repository for QM1 contains the certificate for QM1 and the public certificate from QM2. The key repository for QM2 contains the certificate for QM2 and the public certificate from QM1.

Procedure

1. Prepare the key repository on each queue manager, according to operating system:
 - ▶ **ALW** On AIX, Linux, and Windows systems.
 - ▶ **z/OS** On z/OS systems.
2. Create a self-signed certificate for each queue manager:
 - ▶ **ALW** On AIX, Linux, and Windows systems.
 - ▶ **z/OS** On z/OS systems.
3. Extract a copy of each certificate:
 - ▶ **ALW** On AIX, Linux, and Windows systems.
 - ▶ **z/OS** On z/OS systems.
4. Transfer the public part of the QM1 certificate to the QM2 system and vice versa, using a utility such as FTP ▶ **z/OS**, as described in [Exchanging self-signed certificates](#).
5. Add the partner certificate to the key repository for each queue manager:
 - ▶ **ALW** On AIX, Linux, and Windows systems.
 - ▶ **z/OS** On z/OS systems.
6. On QM1, define a sender channel and associated transmission queue, by issuing commands like the following example:

```

DEFINE CHANNEL(QM1.TO.QM2) CHLTYPE(SDR) TRPTYPE(TCP) CONNAME(QM1.MACH.COM) XMITQ(QM2)
SSLCIPH(TLS_RSA_WITH_AES_128_CBC_SHA) DESCR('Sender channel using TLS from QM1 to QM2')

DEFINE QLOCAL(QM2) USAGE(XMITQ)

```

This example uses CipherSpec TLS_RSA. The CipherSpecs at each end of the channel must be the same.

7. On QM2, define a receiver channel, by issuing a command like the following example:

```
DEFINE CHANNEL(QM1.TO.QM2) CHLTYPE(RCVR) TRPTYPE(TCP) SSLCIPH(TLS_RSA_WITH_AES_128_CBC_SHA)
SSLCAUTH(REQUIRED) DESCR('Receiver channel using TLS from QM1 to QM2')
```

The channel must have the same name as the sender channel you defined in Step “6” on page 100, and use the same CipherSpec.

8. Start the channel.

 On z/OS, see [Starting the sender channel](#).

Results

Key repositories and channels are created as illustrated in [Figure 16 on page 100](#)

What to do next

Check that the task has been completed successfully by using DISPLAY commands. If the task was successful, the resulting output is similar to that shown in the following examples.

From queue manager QM1, enter the following command:

```
DISPLAY CHS(QM1.TO.QM2) SSLPEER SSLCERTI
```

The resulting output is like the following example:

```
DISPLAY CHSTATUS(QM1.TO.QM2) SSLPEER SSLCERTI
  4 : DISPLAY CHSTATUS(QM1.TO.QM2) SSLPEER SSLCERTI
AMQ8417: Display Channel Status details.
CHANNEL(QM1.TO.QM2)                CHLTYPE(SDR)
CONNAME(9.20.25.40)                CURRENT
RQMNAME(QM2)
SSLCERTI("CN=QM2,OU=<Department>,O=<Organization>,ST=<State>,C=<Country>")

SSLPEER("SERIALNUMBER=4C:D0:49:D5:02:5E:02,CN=QM2,OU=<Department>,O=<Organization>,ST=<State>,C=
<Country>")
STATUS(RUNNING)                    SUBSTATE(MQGET)
XMITQ(QM2)
```

From queue manager QM2, enter the following command:

```
DISPLAY CHS(QM1.TO.QM2) SSLPEER SSLCERTI
```

The resulting output is like the following example:

```
DISPLAY CHSTATUS(QM1.TO.QM2) SSLPEER SSLCERTI
  5 : DISPLAY CHSTATUS(QM1.TO.QM2) SSLPEER SSLCERTI
AMQ8417: Display Channel Status details.
CHANNEL(QM2.TO.QM1)                CHLTYPE(RCVR)
CONNAME(9.20.35.92)                CURRENT
RQMNAME(QM1)
SSLCERTI("CN=QM1,OU=<Department>,O=<Organization>,ST=<State>,C=<Country>")

SSLPEER("SERIALNUMBER=4C:D0:49:D5:02:5F:38,CN=QM1,OU=<Department>,O=<Organization>,ST=<State>,C=
<Country>")
STATUS(RUNNING)                    SUBSTATE(RECEIVE)
XMITQ( )
```

In each case, the value of SSLPEER must match that of the DN in the partner certificate that was created in Step “2” on page 100. The issuers name matches the peer name because the certificate is self-signed.

SSLPEER is optional. If it is specified, its value must be set so that the DN in the partner certificate (created in Step “2” on page 100) is allowed. For more information about the use of SSLPEER, see [IBM MQ rules for SSLPEER values](#).

Using CA-signed certificates for mutual authentication of two queue managers

Follow these sample instructions to implement mutual authentication between two queue managers, using CA-signed TLS certificates.

About this task

Scenario:

- You have two queue managers called QM1 and QM2, which need to communicate securely. You require mutual authentication to be carried out between QM1 and QM2.
- In the future you are planning to use this network in a production environment, and therefore you have decided to use CA-signed certificates from the beginning.

The resulting configuration looks like this:

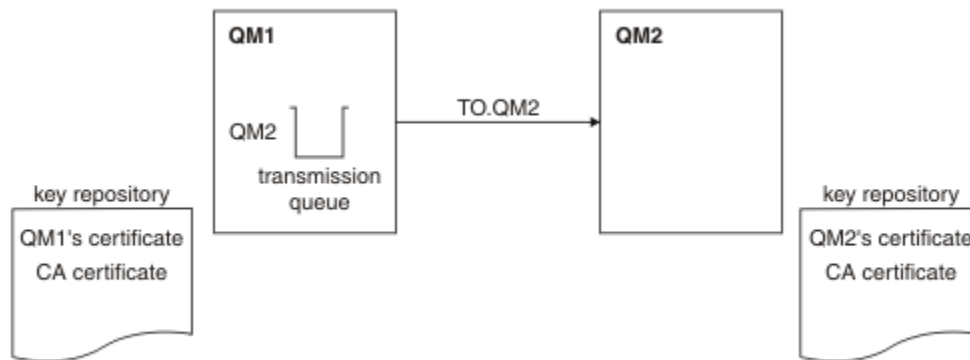


Figure 17. Configuration resulting from this task




In Figure 17 on page 102, the key repository for QM1 contains QM1's certificate and the CA certificate. The key repository for QM2 contains QM2's certificate and the CA certificate.

In this example both QM1's certificate and QM2's certificate were issued by the same CA. If QM1's certificate and QM2's certificate were issued by different CAs then the key repositories for QM1 and QM2 must contain both CA certificates.




Procedure

1. Prepare the key repository on each queue manager, according to the operating system, or systems, your enterprise uses:
 - **IBM i** On IBM i systems.
 - **ALW** On AIX, Linux, and Windows systems.
 - **z/OS** On z/OS systems.
2. Request a CA-signed certificate for each queue manager.
You might use different CAs for the two queue managers.
 - **IBM i** On IBM i systems.
 - **ALW** On AIX, Linux, and Windows systems.
 - **z/OS** On z/OS systems.
3. Add the Certificate Authority certificate to the key repository for each queue manager:

If the Queue managers are using different Certificate Authorities then the CA certificate for each Certificate Authority must be added to both key repositories.

-  Do not perform this step on IBM i systems.
-  On AIX, Linux, and Windows systems.
-  On z/OS systems.

4. Receive the CA-signed certificate to the key repository for each queue manager:

-  On IBM i systems.
-  On AIX, Linux, and Windows systems.
-  On z/OS systems.

5. On QM1, define a sender channel and associated transmission queue by issuing commands like the following example:

```
DEFINE CHANNEL(TO.QM2) CHLTYPE(SDR) TRPTYPE(TCP)
CONNAME(QM2.MACH.COM) XMITQ(QM2) SSLCIPH(TLS_RSA_WITH_AES_128_CBC_SHA256)
DESCR('Sender channel using TLS from QM1 to QM2')

DEFINE QLOCAL(QM2) USAGE(XMITQ)
```




This example uses CipherSpec TLS_RSA_WITH_AES_128_CBC_SHA256. The CipherSpecs at each end of the channel must be the same.

6. On QM2, define a receiver channel by issuing a command like the following example:

```
DEFINE CHANNEL(TO.QM2) CHLTYPE(RCVR) TRPTYPE(TCP)
SSLCIPH(TLS_RSA_WITH_AES_128_CBC_SHA256) SSLCAUTH(REQUIRED)
DESCR('Receiver channel using TLS to QM2')
```

The channel must have the same name as the sender channel you defined in Step “5” on page 103, and use the same CipherSpec.

7. Start the channel:

-  On IBM i systems.
-  On AIX, Linux, and Windows systems.
-  On z/OS systems.

Results

Key repositories and channels are created as illustrated in [Figure 17 on page 102](#).

What to do next

Check that the task has been completed successfully by using DISPLAY commands. If the task was successful, the resulting output is like that shown in the following examples.

From queue manager QM1, enter the following command:

```
DISPLAY CHS(TO.QM2) SSLPEER SSLCERTI
```

The resulting output is like the following example:

```
DISPLAY CHSTATUS(TO.QM2) SSLPEER SSLCERTI
  4 : DISPLAY CHSTATUS(TO.QM2) SSLPEER SSLCERTI
AMQ8417: Display Channel Status details.
CHANNEL(TO.QM2)                                CHLTYPE(SDR)
```

```

CONNAME(192.0.0.2)                                CURRENT
RQMNAME(QM2)
SSLCERTI("CN=<Division> CA,OU=<Department>,O=<Organization>,ST=<State>,C=<Country>")

SSLPEER("SERIALNUMBER=4C:D0:49:D5:02:5F:38,CN=QM2,OU=<Department>,O=<Organization>,ST=<State>,C=
<Country>")
STATUS(RUNNING)                                SUBSTATE(MQGET)
XMITQ(QM2)

```

From the queue manager QM2, enter the following command:

```
DISPLAY CHS(TO.QM2) SSLPEER SSLCERTI
```

The resulting output is like the following example:

```

DISPLAY CHSTATUS(TO.QM2) SSLPEER SSLCERTI
5 : DISPLAY CHSTATUS(TO.QM2) SSLPEER SSLCERTI
AMQ8417: Display Channel Status details.
CHANNEL(TO.QM2)                                CHLTYPE(RCVR)
CONNAME(192.0.0.1)                                CURRENT
RQMNAME(QM1)
SSLCERTI("CN=<Division> CA,OU=<Department>,O=<Organization>,ST=<State>,C=<Country>")

SSLPEER("SERIALNUMBER=4C:D0:49:D5:02:5F:38,CN=QM1,OU=<Department>,O=<Organization>,ST=<State>,C=
<Country>")
STATUS(RUNNING)                                SUBSTATE(RECEIVE)
XMITQ( )

```

In each case, the value of SSLPEER must match that of the Distinguished Name (DN) in the partner certificate that was created in Step “2” on page 102. The issuer name matches the subject DN of the CA certificate that signed the personal certificate added in Step “4” on page 103.

Connecting two queue managers using one-way authentication

Follow these sample instructions to modify a system with mutual authentication to allow a queue manager to connect using one-way authentication to another; that is, when the SSL/TLS client does not send a certificate.

About this task

Scenario:

- Your two queue managers (QM1 and QM2) have been set up as in “Using CA-signed certificates for mutual authentication of two queue managers” on page 102.
- You want to change QM1 so that it connects using one-way authentication to QM2.

The resulting configuration looks like this:

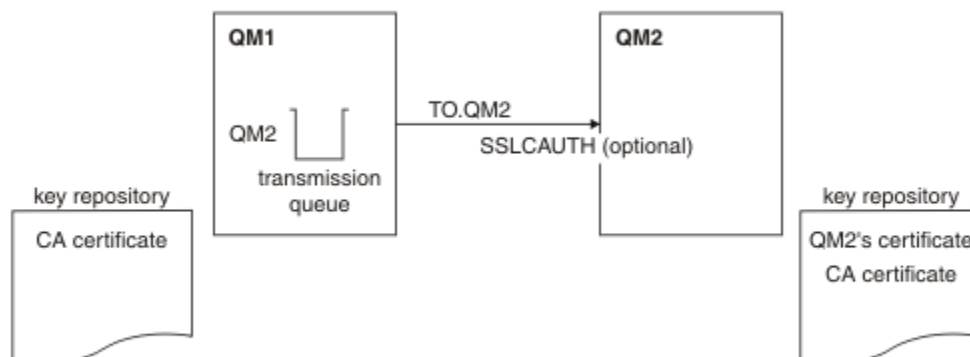





Figure 18. Queue managers allowing one-way authentication

Procedure

1. Remove the personal certificate of QM1 from its key repository:

-  [Removing a certificate on IBM i systems.](#)
-  [Removing a certificate on AIX, Linux, and Windows.](#)
-  [Removing a certificate on z/OS systems.](#) Perform this step twice, to remove both the personal certificate for QMA, and the default certificate.

For details of how certificates are labeled, see [Digital certificate labels](#).

2. Optional: On QM1, if any SSL/TLS channels have run previously, refresh the SSL/TLS environment , as described in [Refreshing the TLS environment](#) .
3. Allow anonymous connections on the receiver , as described in [Allowing anonymous connections on a receiver channel](#) .

Key repositories and channels are changed as illustrated in [Figure 18 on page 104](#)

4. If the sender channel was not running, start it.

Note: If the sender channel was running and you issued the REFRESH SECURITY TYPE(SSL) command (in step 2), the channel restarts automatically.

At the server end of the channel, the presence of the peer name parameter value on the channel status display indicates that a client certificate has flowed.

5. Verify that the task has been completed successfully by issuing some DISPLAY commands.

If the task was successful, the resulting output is similar to that shown in the following examples:

- From the QM1 queue manager, enter the following command:

```
DISPLAY CHS(TO.QM2) SSLPEER SSLCERTI
```

The resulting output will be similar to the following example:

```
DISPLAY CHSTATUS(TO.QMB) SSLPEER SSLCERTI
  4 : DISPLAY CHSTATUS(TO.QMB) SSLPEER
AMQ8417: Display Channel Status details.
CHANNEL(TO.QM2)                                CHLTYPE(SDR)
CONNAME(192.0.0.1)                             CURRENT
RQMNAME(QM2)
SSLCERTI("CN=IBM MQ CA,OU=IBM MQ Devt,O=IBM,ST=Hampshire,C=UK")
SSLPEER("SERIALNUMBER=4C:D0:49:D5:02:5F:38,CN=QMB,OU=IBM MQ
Development,O=IBM,ST=Hampshire,C=UK")
STATUS(RUNNING)                                SUBSTATE(MQGET)
XMITQ(QM2)
```

- From the QM2 queue manager, enter the following command:

```
DISPLAY CHS(TO.QM2) SSLPEER SSLCERTI
```

The resulting output will be similar to the following example:

```
DISPLAY CHSTATUS(TO.QM2) SSLPEER SSLCERTI
  5 : DISPLAY CHSTATUS(TO.QM2) SSLPEER SSLCERTI
AMQ8417: Display Channel Status details.
CHANNEL(TO.QM2)                                CHLTYPE(RCVR)
CONNAME(192.0.0.2)                             CURRENT
RQMNAME(QMA)                                   SSLCERTI( )
SSLPEER( )                                     STATUS(RUNNING)
SUBSTATE(RECEIVE)                             XMITQ( )
```




On QM2, the SSLPEER field is empty, showing that QM1 did not send a certificate. On QM1, the value of SSLPEER matches that of the DN in QM2's personal certificate.

Connecting a client to a queue manager securely

Secure communications that use the TLS cryptographic security protocols involve setting up the communication channels and managing the digital certificates that you will use for authentication.



To set up your SSL/TLS installation you must define your channels to use TLS. You must also obtain and manage your digital certificates. On a test system, you can use self-signed certificates or certificates issued by a local certificate authority (CA). On a production system, do not use self-signed certificates.

For full information about creating and managing certificates, see the following topics:

-  [Working with SSL or TLS on IBM i](#)
-  [Working with SSL or TLS on AIX, Linux, and Windows systems](#)
-  [Working with SSL or TLS on z/OS](#)

This collection of topics introduces the tasks involved in setting up SSL/TLS communications, and provides step-by-step guidance on completing those tasks.

You might also want to test SSL/TLS client authentication, which are an optional part of the protocols. During the SSL/TLS handshake, the SSL/TLS client always obtains and validates a digital certificate from the server. With the IBM MQ implementation, the SSL/TLS server always requests a certificate from the client.

  On IBM i, AIX, Linux, and Windows systems, the SSL/TLS client sends a certificate only if it has one labeled in the correct IBM MQ format, which is either `ibmwebsphereemq` followed by your logon user ID in lowercase, or the value of the **CERTLABL** attribute. See [Digital certificate labels](#).

The SSL/TLS server always validates the client certificate if one is sent. If the client does not send a certificate, authentication fails only if the end of the channel that is acting as the SSL/TLS server is defined with either the **SSLCAUTH** parameter set to `REQUIRED` or an **SSLPEER** parameter value set. For more information about connecting a queue manager anonymously, see [“Connecting a client to a queue manager anonymously” on page 110](#).

Related concepts

[TLS CipherSpecs and CipherSuites in IBM MQ classes for Java](#)

[TLS CipherSpecs and CipherSuites in IBM MQ classes for JMS](#)


Related tasks

[Using certificates for the managed .NET client](#)

Using self-signed certificates for mutual authentication of a client and queue manager

Follow these sample instructions to implement mutual authentication between a client and a queue manager, by using self-signed TLS certificates.

About this task

 DCM on IBM i does not support self-signed certificates, so this task is not applicable on IBM i systems.

Scenario:

- You have a client, C1, and a queue manager, QM1, which need to communicate securely. You require mutual authentication to be carried out between C1 and QM1.
- You have decided to test your secure communication by using self-signed certificates.

The resulting configuration looks like this:

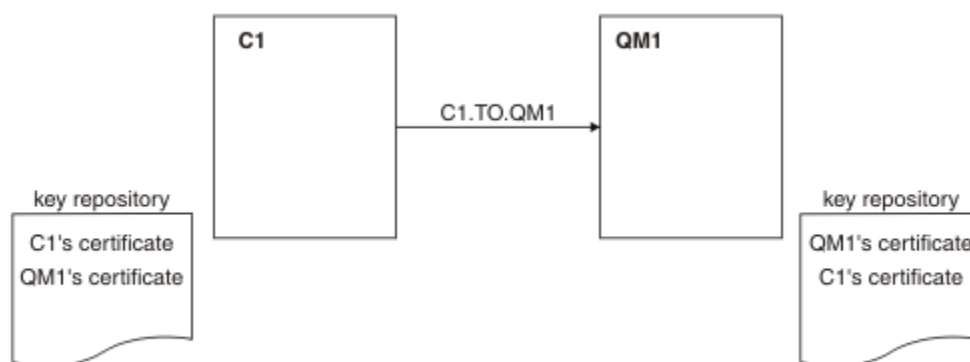


Figure 19. Configuration resulting from this task

In Figure 19 on page 107, the key repository for QM1 contains the certificate for QM1 and the public certificate from C1. The key repository for C1 contains the certificate for C1 and the public certificate from QM1.

Procedure

1. Prepare the key repository on the client and queue manager, according to operating system:
 - **ALW** On AIX, Linux, and Windows systems.
 - **z/OS** On z/OS systems (queue manager only).
2. Create self-signed certificates for the client and queue manager:
 - **ALW** On AIX, Linux, and Windows systems.
 - **z/OS** On z/OS systems (queue manager only).
3. Extract a copy of each certificate:
 - **ALW** On AIX, Linux, and Windows systems.
 - **z/OS** On z/OS systems.
4. Transfer the public part of the C1 certificate to the QM1 system and vice versa, using a utility such as FTP.
 - **z/OS** For z/OS, see [Exchanging self-signed certificates](#).
5. Add the partner certificate to the key repository for the client and queue manager:
 - **ALW** On AIX, Linux, and Windows systems.
 - **z/OS** On z/OS systems.
6. Issue the command `REFRESH SECURITY TYPE(SSL)` on the queue manager.
7. Define a client-connection channel in either of the following ways:
 - Using the `MQCONN` call with the `MQSCO` structure on C1, as described in [Creating a client-connection channel on the IBM MQ MQI client using MQCNO](#).
 - Using a client channel definition table, as described in [Creating server-connection and client-connection definitions on the server](#).
8. On QM1, define a server-connection channel, by issuing a command like the following example:

```
DEFINE CHANNEL(C1.TO.QM1) CHLTYPE(SVRCONN) TRPTYPE(TCP) SSLCIPH(TLS_RSA_WITH_AES_128_CBC_SHA)
SSLCAUTH(REQUIRED) DESCR('Receiver channel using TLS from C1 to QM1')
```

The channel must have the same name as the client-connection channel you defined in step 6, and use the same CipherSpec.

Results

Key repositories and channels are created as illustrated in [Figure 19 on page 107](#).

What to do next

Check that the task has been completed successfully by using **DISPLAY** commands. If the task was successful, the resulting output is similar to that shown in the following example.

From queue manager QM1, enter the following command:

```
DISPLAY CHSTATUS(C1.TO.QM1) SSLPEER SSLCERTI
```

The resulting output is like the following example:

```
DISPLAY CHSTATUS(C1.TO.QM1) SSLPEER SSLCERTI
 5 : DISPLAY CHSTATUS(C1.TO.QM1) SSLPEER SSLCERTI
AMQ8417: Display Channel Status details.
CHANNEL(C1.TO.QM1)                CHLTYPE(SVRCONN)
CONNAME(192.0.0.1)                CURRENT
SSLCERTI("CN=QM1,OU=IBM MQ Development,O=IBM,ST=Hampshire,C=UK")
SSLPEER("SERIALNUMBER=4C:D0:49:D5:02:5E:02,CN=QM2,OU=IBM MQ
Development,O=IBM,ST=Hampshire,C=UK")
STATUS(RUNNING)                   SUBSTATE(RECEIVE)
```

It is optional to set the **SSLPEER** filter attribute of the channel definitions. If the channel definition **SSLPEER** is set, its value must match the subject DN in the partner certificate that was created in Step 2. After a successful connection, the **SSLPEER** field in the **DISPLAY CHSTATUS** output shows the subject DN of the remote client certificate.

Using CA-signed certificates for mutual authentication of a client and queue manager

Follow these sample instructions to implement mutual authentication between a client and a queue manager, by using CA-signed TLS certificates.

About this task

Scenario:

- You have a client, C1, and a queue manager, QM1, which need to communicate securely. You require mutual authentication to be carried out between C1 and QM1.
- In the future you are planning to use this network in a production environment, and therefore you have decided to use CA-signed certificates from the beginning.

The resulting configuration looks like this:

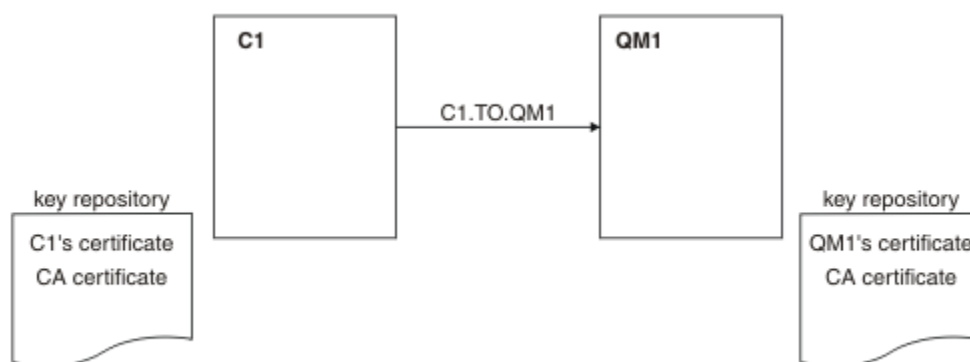


Figure 20. Configuration resulting from this task

In Figure 20 on page 109, the key repository for C1 contains certificate for C1 and the CA certificate. The key repository for QM1 contains the certificate for QM1 and the CA certificate. In this example both C1's certificate and QM1's certificate were issued by the same CA. If C1's certificate and QM1's certificate were issued by different CAs then the key repositories for C1 and QM1 must contain both CA certificates.

Procedure

1. Prepare the key repository on the client and queue manager, according to operating system:

- **IBM i** [On IBM i systems.](#)
- **ALW** [On AIX, Linux, and Windows systems.](#)
- **z/OS** [On z/OS systems \(queue manager only\).](#)

2. Request a CA-signed certificate for the client and queue manager.

You might use different CAs for the client and queue manager.

- **IBM i** [On IBM i systems.](#)
- **ALW** [On AIX, Linux, and Windows systems.](#)
- **z/OS** [On z/OS systems \(queue manager only\).](#)

3. Add the certificate authority certificate to the key repository for the client and queue manager.

If the client and queue manager are using different Certificate Authorities then the CA certificate for each Certificate Authority must be added to both key repositories.

- **IBM i** [Do not perform this step on IBM i systems.](#)
- **ALW** [On AIX, Linux, and Windows systems.](#)
- **z/OS** [On z/OS systems \(queue manager only\).](#)

4. Receive the CA-signed certificate to the key repository for the client and queue manager:

- **IBM i** [On IBM i systems.](#)
- **ALW** [On AIX, Linux, and Windows systems.](#)
- **z/OS** [On z/OS systems \(queue manager only\).](#)

5. Define a client-connection channel in either of the following ways:

- Using the MQCONN call with the MQSCO structure on C1, as described in [Creating a client-connection channel on the IBM MQ MQI client using MQCNO.](#)

- Using a client channel definition table, as described in [Creating server-connection and client-connection definitions on the server](#).

6. On QM1, define a server-connection channel by issuing a command like the following example:

```
DEFINE CHANNEL(C1.TO.QM1) CHLTYPE(SVRCONN) TRPTYPE(TCP)
SSLCIPH(TLS_RSA_WITH_AES_128_CBC_SHA) SSLCAUTH(REQUIRED)
DESCR('Receiver channel using TLS from C1 to QM1')
```

The channel must have the same name as the client-connection channel you defined in step 6, and use the same CipherSpec.

Results

Key repositories and channels are created as illustrated in [Figure 20 on page 109](#).

What to do next

Check that the task has been completed successfully by using DISPLAY commands. If the task was successful, the resulting output is like that shown in the following example.

From the queue manager QM1, enter the following command:

```
DISPLAY CHSTATUS(TO.QMB) SSLPEER SSLCERTI
```

The resulting output is like the following example:

```
DISPLAY CHSTATUS(C1.TO.QM1) SSLPEER SSLCERTI
5 : DISPLAY CHSTATUS(C1.TO.QM1) SSLPEER SSLCERTI
AMQ8417: Display Channel Status details.
CHANNEL(C1.TO.QM1)                CHLTYPE(SVRCONN)
CONNAME(192.0.0.1)                CURRENT
SSLCERTI("CN=IBM MQ CA,OU=IBM MQ Devt,O=IBM,ST=Hampshire,C=UK")
SSLPEER("SERIALNUMBER=4C:D0:49:D5:02:5F:38,CN=QMA,OU=IBM MQ
Development,O=IBM,ST=Hampshire,C=UK")
STATUS(RUNNING)                   SUBSTATE(RECEIVE)
```

The SSLPEER field in the DISPLAY CHSTATUS output shows the subject DN of the remote client certificate that was created in Step 2. The issuer name matches the subject DN of the CA certificate that signed the personal certificate added in Step 4.

Connecting a client to a queue manager anonymously

Follow these sample instructions to modify a system with mutual authentication to allow a queue manager to connect anonymously to another.

About this task

Scenario:

- Your queue manager and client (QM1 and C1) have been set up as in [“Using CA-signed certificates for mutual authentication of a client and queue manager” on page 108](#).
- You want to change C1 so that it connects anonymously to QM1.

The resulting configuration looks like this:

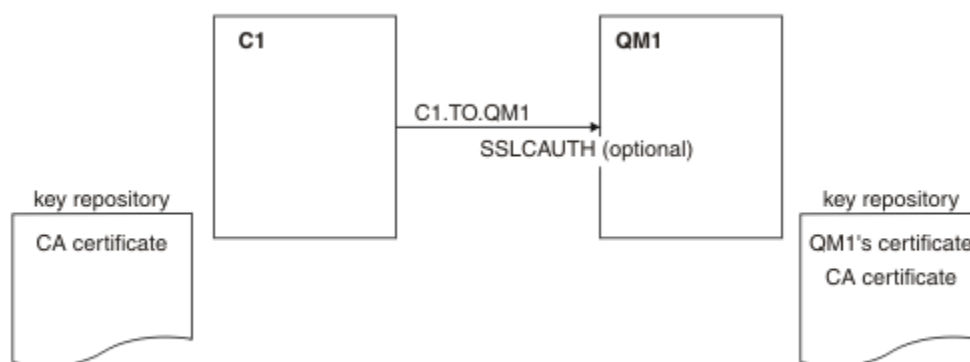


Figure 21. Client and queue manager allowing anonymous connection

Procedure

1. Remove the personal certificate from the key repository for C1, according to operating system:

- **IBM i** [IBM i systems.](#)
- **ALW** [AIX, Linux, and Windows systems.](#)

The certificate label is either `ibmwebsphermq` followed by your logon user ID in lowercase, or the value of the **CERTLABL** attribute. See [Digital certificate labels](#).

2. Restart the client application, or cause the client application to close and reopen all SSL/TLS connections.
3. Allow anonymous connections on the queue manager, by issuing the following command:

```
ALTER CHANNEL(C1.TO.QM1) CHLTYPE(SVRCONN) SSLCAUTH(OPTIONAL)
```

Results

Key repositories and channels are changed as illustrated in [Figure 21 on page 111](#)

What to do next

At the server end of the channel, the presence of the peer name parameter value on the channel status display indicates that a client certificate has flowed.

Verify that the task has been completed successfully by issuing some **DISPLAY** commands. If the task was successful, the resulting output is similar to that shown in the following example:

From queue manager QM1, enter the following command:

```
DISPLAY CHSTATUS(C1.TO.QM1) SSLPEER SSLCERTI
```

The resulting output will be similar to the following example:

```
DISPLAY CHSTATUS(C1.TO.QM1) SSLPEER SSLCERTI
5 : DISPLAY CHSTATUS(C1.TO.QM1) SSLPEER SSLCERTI
AMQ8417: Display Channel Status details.
CHANNEL(C1.TO.QM1)          CHLTYPE(SVRCONN)
CONNAME(192.0.0.1)          CURRENT
SSLCERTI( )                 SSLPEER( )
STATUS(RUNNING)             SUBSTATE(RECEIVE)
```

The **SSLCERTI** and **SSLPEER** fields are empty, showing that C1 did not send a certificate.

Windows Migrating on Windows

Starting with an existing IBM MQ 9.3 installation, this scenario leads you through the key tasks required to upgrade and migrate data to IBM MQ 9.4. Both versions are installed on the same Windows environment.

Windows Planning the solution

Review the topics in this section to understand what is covered in this scenario, the reasons why a business might want to follow the scenario, and an overview of the solution proposed by the scenario.

Related tasks

[Planning your installation](#)

Windows Assumptions

This scenario makes several assumptions about the system that you are using to set up and work with the sample IT configuration. These assumptions include the operating system and version of the products that you are using, and whether or not you have security configured for IBM MQ.

This scenario assumes the following points:

- You are using one computer with a Windows operating system for this scenario, on which you will install both the initial IBM MQ 9.3 configuration and then IBM MQ 9.4.

Note: Clustering is not described in this scenario. Instructions are provided for installing a sample IBM MQ single server configuration which you can use as a starting point for trying out the scenario in the same way as it was originally developed.

- You are using the following versions of IBM MQ:
 - For the initial sample configuration, you are using IBM MQ 9.3.
 - For the post-migration configuration, you are using IBM MQ 9.4.
- This scenario does not describe the security configuration for IBM MQ. If you do have security configured you should still be able to complete the scenario.
- You can use the Windows command prompt, and the graphical user interface IBM MQ Explorer, to complete the tasks described in this scenario.

Related concepts

[Migration paths](#)

Windows Business overview

A company wants to migrate the existing IBM MQ 9.3 IT configuration on a Windows operating system to IBM MQ 9.4.

The company decides to migrate their business solution to IBM MQ 9.4 to gain business value including:

- Using new and updated functionality available in IBM MQ 9.4.
- Exploring the new type of release available from IBM MQ 9.4; Continuous Delivery Release (CDR).
- Benefiting from LDAP authorisation on Windows platforms.

Related concepts

[IBM MQ release types and versioning](#)

Related information

[IBM MQ FAQ for Long Term Support and Continuous Delivery releases](#)

When you are migrating between IBM MQ 9.3 and IBM MQ 9.4, there are several migration paths you can take.

This topic gives an overview of the following migration paths:

- Single-stage path, also known as Stand-alone, migration
- Side-by-side migration path
- Multi-stage migration path

Note: This scenario describes only the Single-stage and Side-by-side migration methods.

Consider the advantages and limitations of each path to determine which best suits your requirements:

Single-stage migration

In single-stage migration, the installation of the latest version of the product replaces an earlier version in the same installation location.

The advantage of single-stage migration is that it changes the configuration of a queue manager on the earlier version as little as possible. Existing applications switch from loading the libraries from the earlier version, to loading the libraries of the latest version, automatically. Using this approach, your system will be unavailable for the entire process.

Side-by-side migration

In side-by-side migration, you install the latest version of IBM MQ alongside queue managers that continue to be associated with the earlier version.

When you are ready, you migrate the queue managers, and applications to the latest version.

With this approach, because you uninstall the earlier version before starting any queue managers, you can assign an installation of the latest version to be the primary installation.

See [Choosing a primary installation](#) for further information.

Multi-stage migration

In multi-stage migration, you install the latest version of the product alongside running queue managers that continue to be associated with the earlier version. You can create queue managers and run new applications using the latest version installation. When you are ready to start migrating queue managers and applications from the earlier, you can do so, one-by-one. When migration to the latest version is complete, uninstall the earlier version, and make the latest version installation the primary installation.

With the multi-stage approach, until you uninstall the earlier version, you must configure an environment to run applications that connect to a queue manager to the latest version. You must also provide a path to run IBM MQ commands. Both these tasks are accomplished with the [`setmqenv`](#) command.

Related concepts

[Overview of migration concepts and methods](#)

Related tasks

[Migrating on AIX and Linux: single-stage](#)

[Migrating on AIX and Linux: side-by-side](#)

[Migrating on AIX and Linux: multi-stage](#)

Related reference

[List of changes that affect migration](#)

Windows Technical solution

This scenario describes two methods of migrating from an earlier version of IBM MQ to a later version, where both versions run on a Windows operating system and are on the same server.

Windows Overview: Initial IT configuration

A company uses an existing IT configuration provided by IBM MQ 9.3 installed on a server with a Windows operating system. This scenario describes migrating the initial IT configuration to an equivalent IT configuration provided by IBM MQ 9.4 on the same server.

The initial IT configuration includes several components that an administrator configures or uses, as shown in [Figure 22 on page 114](#):

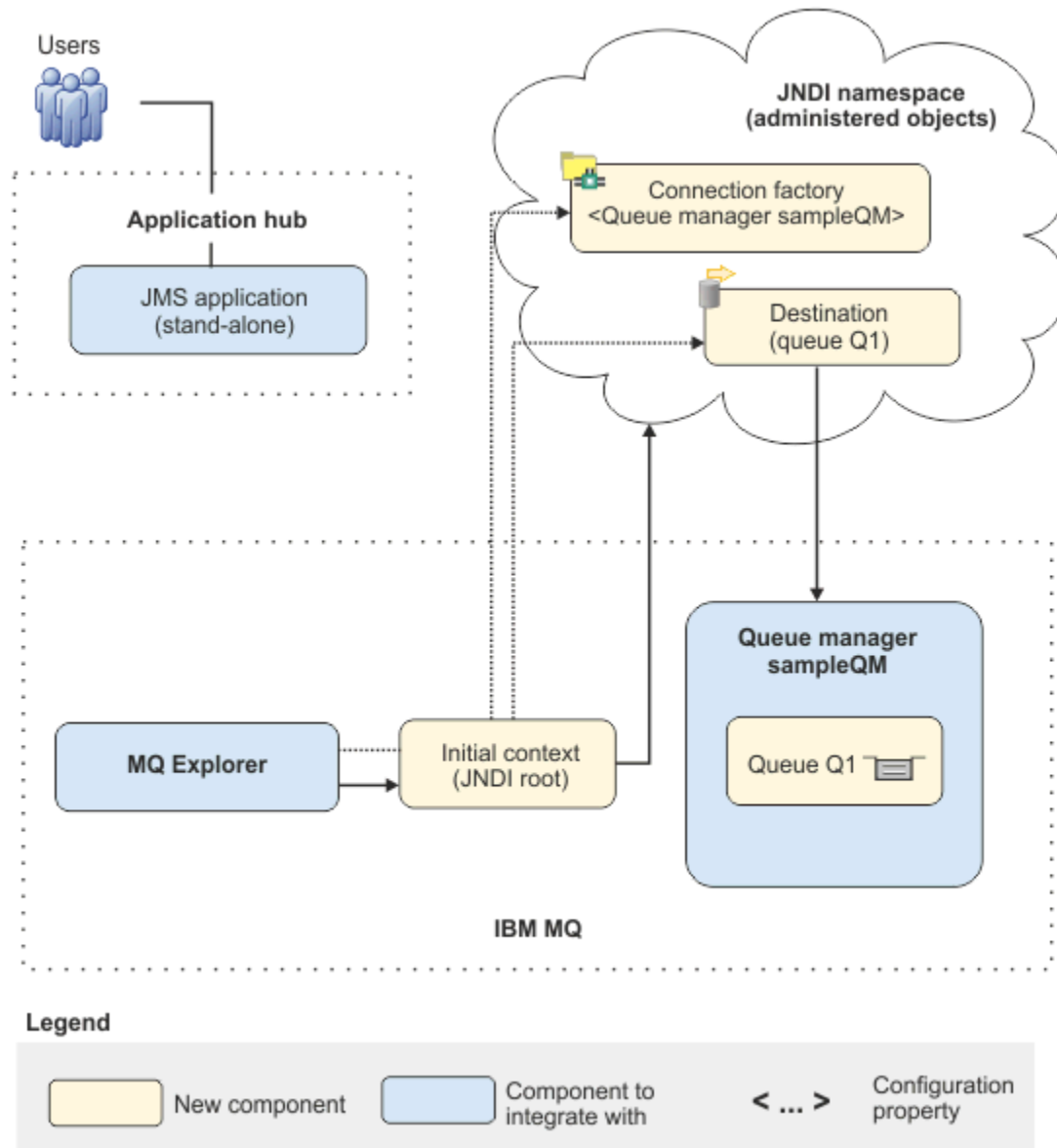


Figure 22. Initial IT configuration

JMS application

A stand-alone application that business users interact with, for example to register an order. The application uses the Java Message Service (JMS) for asynchronous messaging.

- JMS is the Java EE messaging standard that is widely supported. JMS-based applications are therefore portable across many messaging products.
- JMS provides a level of abstraction from the details of the messaging layer, simplifying the application development process.
- JMS provides asynchronous communication, enabling applications to run without having to wait for a reply, unlike tightly coupled systems such as remote procedure call (RPC).
- Applications that use JMS do not directly specify details to access resources. Instead, they look up and use administered JMS objects such as a connection factory and a destination.

For some situations, other messaging standards might be more suitable than JMS. For example IBM Message Service Clients for C, C++ and .NET, also known as XMS, are APIs that provide similar benefits to JMS for non-Java applications. XMS is therefore more suitable if you are using the .NET platform, or you want to integrate existing C++ applications with newer Java EE applications.

The application uses point-to-point messaging to send messages to a queue in the infrastructure and processes reply messages, to provide the business user with an appropriate response.

In this messaging model, an application sends a message to a queue, and another application receives the message from the queue and acknowledges receipt of the message. This model is the simplest form of messaging because it involves only two endpoints. This model is also the most appropriate for the scenario sample application: a single client requests information from a single server.

In the alternative messaging model, publish/subscribe, a publisher publishes a message to a message topic. Subscribers subscribe to the topic to receive messages. The publisher and subscriber do not have any information about each other, and the message is received by zero or more recipients.

Queue manager sampleQM

The IBM MQ queue manager that provides the initial messaging infrastructure. It hosts the queue that the JMS application works with.

Q1 [Message queue]

The IBM MQ queue that the JMS application sends messages to.

JNDI namespace

A Java Naming Directory Interface JNDI namespace is used to hold JMS administered objects, which applications can use to connect to IBM MQ and access destinations to send or receive messages.

JNDI is part of Java EE, and provides a standard way for applications to access various types of naming and directory services, for the retrieval of application components. For example, you can use JNDI to access a naming service on a file system to retrieve the location of a printer object, or to access a directory service on an LDAP server to retrieve a user object which contains ID and password information. JNDI therefore enhances the portability of JMS-based applications, and makes it easier to integrate those applications with each other and into existing systems. For JMS messaging, you use JNDI to store objects that represent the target destination of a message, or the connection factory that creates the connection between your application and its messaging destination.

Any application or process with access to the JNDI namespace can use the same administered objects. The properties of the administered objects can be changed in JNDI, with all the applications or processes able to benefit from those same changes.

Initial context

An initial context defines the root of the JNDI namespace. To use IBM MQ Explorer to create and configure administered objects, you first add an initial context that defines the root of the JNDI namespace. Similarly, a JMS application first obtains an initial context, before it can retrieve administered objects from the JNDI namespace.

Connection factory, myCF

A JMS connection factory object defines a set of standard configuration properties for connections. An application uses a connection factory to create a connection to IBM MQ.

Destination, myQueue

A JMS destination can be a topic or a queue. In this scenario, the destination is a queue, and identifies the IBM MQ queue that applications send messages to, or from which an application receives messages, or both. An application looks up the destination in the JNDI namespace to create a connection to the IBM MQ queue.

Windows Overview: The delivered logical topology

The company has migrated from IBM MQ 9.3 to IBM MQ 9.4.

The IT configuration following the migration is unchanged, as shown in [Figure 23 on page 116](#). The queue manager and sample application are migrated, and the business can benefit from the new functionality in the latest version of IBM MQ.

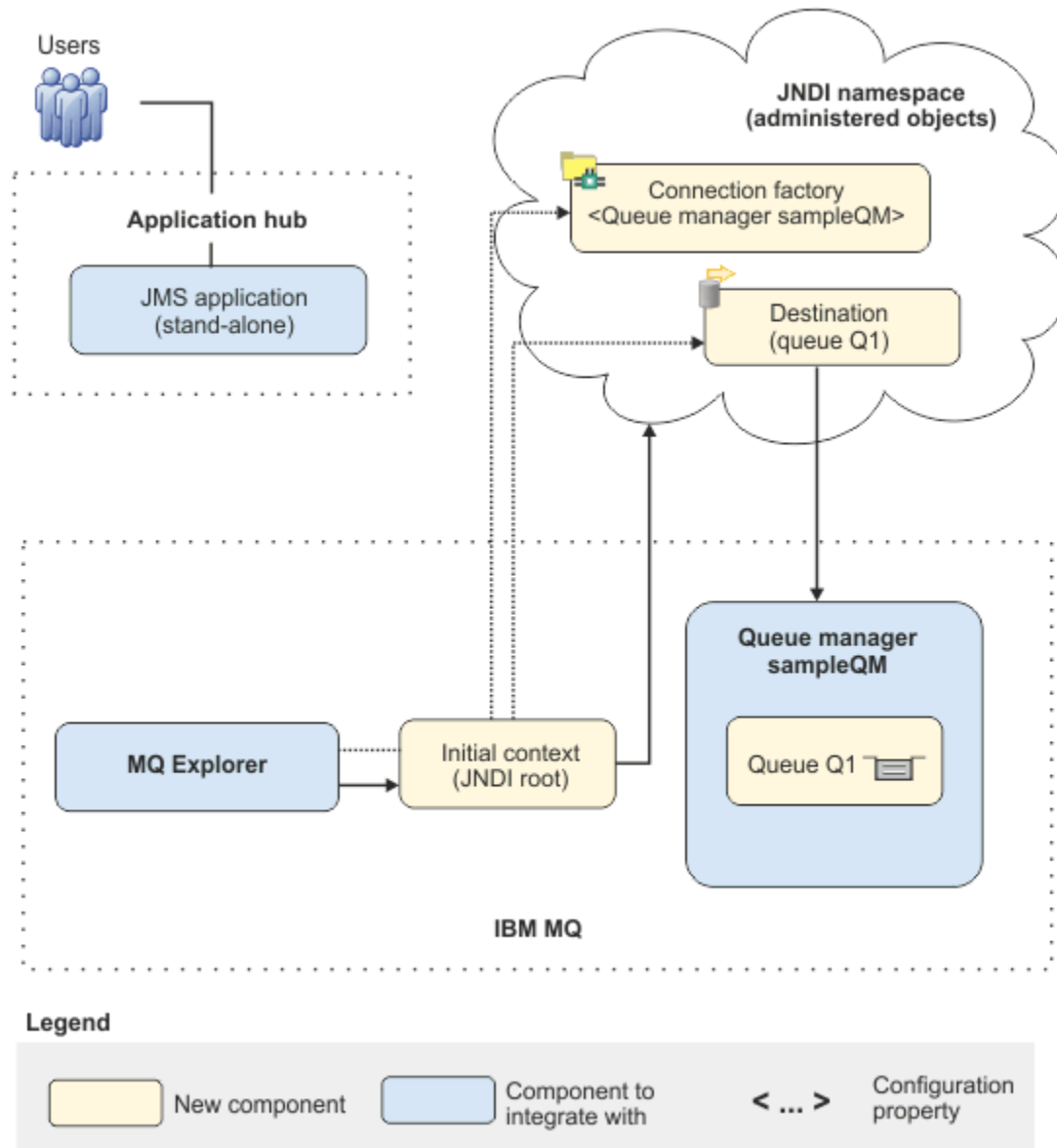


Figure 23. Delivered IT configuration

Windows Implementing the solution using the graphical user interface

Implementing the solution in this scenario involves using the graphical user interface to migrate from an earlier release of IBM MQ, running on a Windows operating system, to a later release, also running on a

Windows operating system, then verifying that the queue managers and queues have been successfully migrated to the later release.

Before you begin

If you want to try out the scenario, first follow the instructions to set up a copy of the sample messaging infrastructure as described in [“Creating an initial IT configuration” on page 118](#). This sample configuration is based on IBM MQ 9.3.

About this task

The migration process described in this scenario shows migrating a queue manager from IBM MQ 9.3 to IBM MQ 9.4.

The scenario includes two options for the migration method. You can choose to try out one or the other of these options, or both.

Option 1: Single-stage migration

In single-stage migration, the installation of the later version of the product replaces an earlier version in the same installation location.

The advantage of single-stage migration is that it changes the configuration of a queue manager on the earlier version as little as possible. Existing applications switch from loading the libraries from the earlier version, to loading the libraries of the latest version, automatically. Using this approach, your system will be unavailable for the entire process.

Option 2: Side-by-side migration

In side-by-side migration, you install the later version of IBM MQ alongside an earlier version. The queue managers continue to be associated with the earlier version until you are ready to migrate them to the later version.

With the side-by-side approach, because you uninstall the earlier version before starting any migrated queue managers in the later version, you can assign the installation of the later version to be the primary installation.

Procedure

1. Create a sample initial IT configuration to use as a starting point for the scenario as described in [“Creating an initial IT configuration” on page 118](#).
2. Select the method that you are going to use to migrate the product then follow the instructions for your chosen option:
 - [“Option 1: Single-stage migration” on page 127](#)
 - [“Option 2: Side-by-side migration” on page 133](#)

Related tasks

[Migrating on AIX and Linux: single-stage](#)

[Migrating on AIX and Linux: side-by-side](#)

[Migrating a queue manager from a previous version to the latest version on Windows](#)

[Choosing a primary installation](#)

Creating an initial IT configuration

This scenario was developed using a sample initial (IT) configuration. Follow the instructions to set up this sample configuration to try out the scenario in the same way as it was originally developed.

About this task

The initial IT configuration for this scenario, which is described in [“Overview: Initial IT configuration” on page 114](#), includes an initial context, added for the IBM MQ Explorer to connect to the root of the JNDI namespace. The JNDI namespace includes a connection factory, added for the sample JMS application to use to connect to IBM MQ, and a destination, added for the sample JMS application to connect to the IBM MQ queue. That IBM MQ queue has also been added into the initial IT configuration and is used by the sample JMS application.

Procedure

1. [Install IBM MQ 9.3](#) and verify your installation.
2. [Configure the JNDI namespace and administered objects](#).
3. [Verify the sample IT configuration](#).

Installing IBM MQ 9.4 by using the launchpad

Use the installation launchpad and wizards to install the version of IBM MQ that you want to set up as an initial IT configuration to use as a starting point for this scenario.

Before you begin

Before starting this task, complete the following checks:

- You must have local administrator authority when you are installing.
- Ensure that the machine name does not contain any spaces.
- Ensure that you have sufficient disk space. For more information, see [Disk space requirements on Multiplatforms](#).

For this scenario, it is not necessary to determine whether you need to define Windows domain user ID's for any IBM MQ users, as this requirement is outside the scope of this scenario. See [Creating an Active Directory and DNS domain for IBM MQ](#) for more information.

Before you install IBM MQ, check that your system meets the hardware and software requirements. For the latest details of hardware and software requirements, see [System Requirements for IBM MQ](#).

About this task

This task describes the basic steps for installing IBM MQ on a Windows operating system.

The installation programs contain links to further information. The installation process has the following parts:

1. Start the installation process.
2. Use the Installation Launchpad to check and install software requirements, specify network information, and start the IBM MQ installation wizard.
3. Use the IBM MQ installation wizard to install the software, and start the Prepare IBM MQ Wizard.
4. Use the Prepare IBM MQ Wizard to start the IBM MQ service.

Procedure

1. Start the installation process.

In Windows Explorer, navigate to the temporary folder into which you downloaded the installation image, then double-click `setup.exe`.

The installation launchpad is started.

2. Using the launchpad, review and, if necessary, modify the software requirements and network configuration.

- a) Click the **Software Requirements** button to display the **Software Requirements** tab.
- b) Check that the software requirements have been met and that the entry for the requirement displays a green tick with the words OK. Make any indicated corrections.

Note: To see more details about any requirement, click the plus (+) button.

- c) Click the **Network Configuration** button to display the **Network Configuration** tab.
- d) Click the **No** radio button.

Note: This scenario assumes that you do not need to configure a domain user ID for IBM MQ. For more information regarding configuring IBM MQ for Windows domain users, click the **More information** button.

- e) On the **IBM MQ Installation** tab of the Launchpad, select the installation language, and then click the **Launch IBM MQ Installer** button to start the IBM MQ installation wizard.

You have completed your review IBM MQ of the installation requirements, have made any required modifications, and have started the IBM MQ installation wizard.

3. Use the IBM MQ installation wizard to install the software, and to start the Prepare IBM MQ Wizard.
 - a) In the IBM MQ installation wizard, read the License Agreement and select the **I accept the terms in the license agreement** check box, then click **Next**.
 - b) Click **Typical**, and then click **Next**.
 - c) On the **Ready to Install IBM MQ** page, review the installation information and click **Install**.

Note: the following details:

- Installation Name
- Top-level folder for Program Files
- Top-level folder for Data Files

The following features will be installed:

- IBM MQ Server
- IBM MQ: a graphical interface for administering and monitoring IBM MQ resources
- Java™ and .NET Messaging and Web Services
- IBM MQ Development Toolkit

The installation process begins. Depending on your system, the installation process can take several minutes.

At the end of the installation process, the IBM MQ Setup window displays the message **Installation Wizard Completed Successfully**.

- d) Click **Finish**.

You have successfully installed IBM MQ. The Prepare IBM MQ Wizard starts automatically, displaying the **Welcome to the Prepare IBM MQ Wizard** page.

4. Use the Prepare MQ wizard to start the IBM MQ service.

- a) On the Welcome to the Prepare IBM MQ Wizard page, select **Next**.

The Prepare IBM MQ Wizard displays the message **Status: Checking IBM MQ Configuration** and a progress bar. When the process is complete the IBM MQ Network Configuration page is displayed.

- b) On the IBM MQ Network Configuration page of the Prepare IBM MQ Wizard, select **No**.
- c) Click **Next**.

The Prepare IBM MQ Wizard displays a message **Status: starting the IBM MQ Service**, and a progress bar. When the process is complete the wizard displays the **Completing the Prepare IBM MQ Wizard** page.

- d) Select **Launch IBM MQ Explorer** and choose whether to start Notepad to view the release notes, and then click **Finish**.

IBM MQ Explorer starts.

You have installed IBM MQ, and you have started the IBM MQ Explorer.

Results

IBM MQ is installed on your computer.

What to do next

You are ready to create the administered objects used in this scenario as described in [“Configuring the JNDI namespace and administered objects”](#) on page 120.

Related concepts

[Hardware and software requirements on Windows systems](#)

[Introduction to IBM MQ](#)

Related tasks

[Installing an IBM MQ server on Windows](#)

[Configuring an IBM MQ server](#)

Configuring the JNDI namespace and administered objects

Define an initial context for the JNDI namespace in IBM MQ Explorer, then, in the namespace, define the administered objects that the sample application can use.

About this task

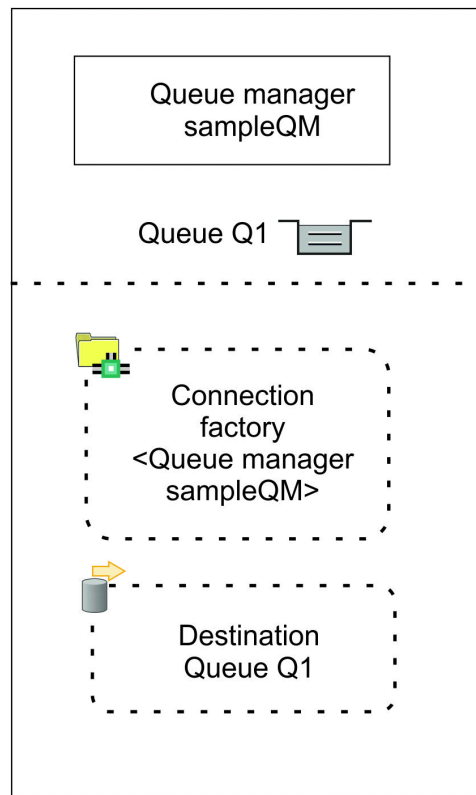
In this task, you create the following objects in IBM MQ.

- A JNDI namespace located in a local file system. A file system is used, because it is the simplest JNDI mechanism for a sample scenario.

The JNDI namespace can be on a file system, Lightweight Directory Access Protocol (LDAP) server, or on another JNDI implementation. If you want to use a JNDI namespace on an LDAP server or another JNDI implementation, you must configure the JNDI namespace and modify the sample application to reference the JNDI namespace, as required by the implementation.

- Administered objects in the JNDI namespace. The JMS application can look up the administered objects to connect to IBM MQ and access IBM MQ destinations with which to send or receive messages.

WebSphere MQ



WebSphere MQ JNDI Namespace

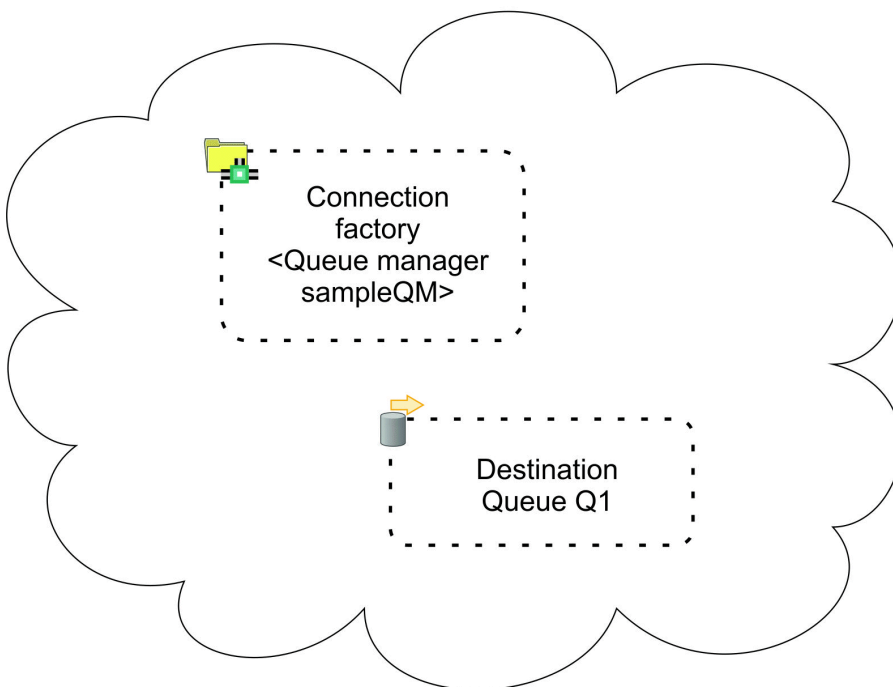


Figure 24. Objects created in IBM MQ

Procedure

1. Start IBM MQ Explorer, if it is not already started, by clicking **Start > All Programs > IBM MQ > IBM MQ Explorer**.

If IBM MQ Explorer is running and displaying the Welcome page, close the Welcome page to start administering IBM MQ objects.

2. Create a queue manager to use for the sample application.

- a) Right click **Queue Managers** and select **New > Queue Manager...**

The Create Queue Manager wizard starts.

- b) In the **Queue manager name** field type sampleQM.

You can choose a different name for the queue manager, but you must remember to use it in later configuration steps, in place of sampleQM.

Note: The name must have no more 48 characters, from the following set:

- Uppercase or lowercase characters (A-Z a-z)
- Numeric characters (0-9)
- Period (.)
- Forward slash (/)
- Underscore (_)
- Percent sign (%)

Names are case-sensitive. Objects of the same type must have different names. For example two queues cannot have the same name, but a queue manager and a queue can.

- c) In the **Dead-letter queue** field type SYSTEM.DEAD.LETTER.QUEUE.

This field is the name of the dead-letter queue that is automatically created when you create the queue manager.

A dead-letter queue stores messages that cannot be delivered to their correct destination, for example because the queue is full. All queue managers should have an associated dead-letter queue.

- d) Leave the other fields empty and click **Finish**, or if that button is disabled, click **Next**.

The **Finish** button is disabled if the port number conflicts with an existing queue manager, for example the queue manager that is created as part of the default configuration. You must continue through the wizard to change the default port number.

- e) If you clicked **Next**, continue to accept the defaults and click **Next** on each page until you get to the final page of the wizard, when the **Finish** button becomes available. Change the specified port number, for example to 1415, and click **Finish**.

IBM MQ displays a dialog window while the queue manager is created and started.

3. Add an initial context for the *JNDI namespace* then connect IBM MQ Explorer to that context

Before you can use IBM MQ Explorer to create and configure JMS administered objects, you must add an initial context to define the root of the JNDI namespace in which the administered objects are stored.

Whenever you want to use IBM MQ Explorer to create or manage administered objects in the JNDI namespace, you must connect IBM MQ Explorer to the initial context of the JNDI namespace.

- a) In the IBM MQ Explorer - **Navigator pane**, right-click **JMS Administered Objects**, then select **Add Initial Context...**

This action displays the " **Connection details** " page.

- b) Under " **Where is the JNDI namespace located?** ", select the **File system** check box.

- c) In the **Bindings directory** field, type C:\JNDI-Directory.

This value matches the JNDI namespace location specified in the sample JMS application. If you must specify a different JNDI directory, you must modify the application to match.

If the directory does not exist on your system the window displays the message Specified location does not exist or is not readable . Click **Browse...** to open a file system window, navigate to Local Disk (C:), then click **Make New Folder** to create the JNDI - Directory folder. Click **OK**.

Click **Next**.

d) On the **User preferences** page, leave the default settings.

- **Context nickname:** The location of the JNDI namespace will be used as the nickname to display the initial context in IBM MQ Explorer.
- **Connect immediately on finish:** This option connects IBM MQ Explorer to the JNDI namespace when you finish creating the initial context, so that you can create administered objects immediately.
- **Automatically reconnect to context on startup:** This option is not selected, because usually you do not need IBM MQ Explorer to automatically reconnect to the initial context every time that you close and reopen IBM MQ Explorer.

If you routinely use IBM MQ Explorer to create or manage administered objects in the JNDI namespace, you can select the **Automatically reconnect to context on startup** check box to cause IBM MQ Explorer to automatically reconnect to the initial context whenever IBM MQ Explorer is started. This option saves you having to manually connect IBM MQ Explorer to the initial context.

Click **Finish** to create and display the initial context.

4. Create a connection factory administered object.

A connection factory administered object defines a set of standard configuration properties for connections. An application uses a connection factory to create a connection to IBM MQ.

a) In the IBM MQ Explorer - **Navigator** pane, expand **JMS Administered Objects**, then expand the initial context, labeled **file:/C:/JNDI-Directory/**.

b) Right-click **Connection Factories**, then select **New > Connection Factory...**

This action displays the **New Connection Factory** wizard

c) In the Name field, type myCF

The sample JMS application contains code that looks up a connection factory with the name myCF. If you must use a different name, you must modify the application to match.

IBM MQ is used for the messaging provider, because the sample application uses point-to-point messaging.

Click **Next**.

d) Leave the type of connection factory as **Connection Factory**, because this option is the most flexible for general JMS use.

A domain-independent connection factory enables JMS applications to use both point-to-point messaging and publish/subscribe messaging, especially if you want the JMS application to perform both types of messaging under the same transaction.

If a JMS application is intended to use only point-to-point messaging or only publish/subscribe messaging, you can select the specific messaging domain when you create the connection factory and a domain-specific (queue or topic) connection factory is created.

e) Leave the support for XA transactions as cleared.

The sample application does not use XA-compliant transactions.

IBM MQ JMS supports XA-compliant transactions in bindings mode. If you want the sample application to use XA-compliant transactions, you must modify the sample application.

Click **Next**.

f) Leave the transport as **Bindings**.

The sample JMS application that uses the connection factory runs on the same computer as the queue manager, so can use Bindings mode transport. This option means that the JMS application

connects directly to the queue manager, and offers a performance advantage over the alternative client mode.

Click **Next**, then **Next** again.

- g) On the **Change properties** page, select **Connection** from the menu on the left, then in the **Connection** pane select sampleQM as the **Base queue manager**.

The base queue manager is the queue manager that the application will connect to. Leave this value blank if you want the application to be able to connect to more than one queue manager.

- h) Click **Finish**.

IBM MQ displays a dialog window to show that the object was created successfully. Click **OK** to close the dialog window.

5. Create a destination administered object.

A destination administered object identifies the IBM MQ queue that applications send messages to, or from which an application receives messages, or both. An application looks up the destination in the JNDI namespace to create a connection to the IBM MQ queue.

In publish/subscribe messaging, the destination identifies a topic rather than a queue.

- a) In the IBM MQ Explorer - **Navigator** pane, expand **JMS Administered Objects**, then expand the initial context, labeled **file:/C:/JNDI-Directory/**.

- b) Right-click **Destinations**, then select **New > Destination....**

The **New Destination** wizard is displayed.

- c) In the **Name** field, type myQueue.

Leave the **Type** as **Queue**.

The sample JMS application contains code that looks up a destination with the name myQueue. The sample JMS application uses point-to-point messaging, so requires a destination of type queue. Destinations of type topic are used for publish/subscribe messaging.

- d) Select the **Start wizard to create a matching MQ Queue** check box.

The destination object needs a matching IBM MQ queue, and it is convenient to use IBM MQ Explorer to create both together. When you complete the **New Destination** wizard, the **Create an MQ Queue** wizard opens, with many of the destination details mapped to the IBM MQ queue.

Click **Next**.

Click **Next** again.

- e) On the "**Change properties**" page, click **Select...** next to **Queue manager**. Select the sampleQM queue manager that you created earlier then click **OK**.

- f) Specify Q1 as the name of the IBM MQ queue.

You can choose a different name for the queue, but you must remember to use it in later configuration steps, in place of Q1.

Note: The name must have no more 48 characters, from the following set:

- Uppercase or lowercase characters (A-Z a-z)
- Numeric characters (0-9)
- Period (.)
- Forward slash (/)
- Underscore (_)
- Percent sign (%)

Names are case-sensitive. Objects of the same type must have different names. For example two queues cannot have the same name, but a queue manager and a queue can.

- g) Click **Finish**.

The **Create an MQ Queue** wizard starts.

If the wizard does not start, you might not have selected the **Start wizard to create a matching MQ Queue** check box in an earlier step. In the IBM MQ Explorer - **Navigator** pane, expand the **sampleQM** queue manager, right click **Queues**, then select **New > Local Queue...**

6. Create a matching IBM MQ queue.

The destination administered object created earlier represents an IBM MQ queue. This queue is where the JMS messages are stored.

- a) Click **Next** to accept the sampleQM queue manager that you specified earlier.
- b) Click **Next**.
- c) Click **Finish** to create the IBM MQ queue using the information from the destination administered object that you created earlier.

IBM MQ displays a dialog window with the message that the object was created successfully.

The new queue is now visible in the **Queues** section under the queue manager.

Results

You have now created the IBM MQ objects that are required to use the sample JMS application.

What to do next

You are now ready to verify that you have configured IBM MQ correctly for use with the sample application as described in [“Verifying the sample IT configuration” on page 125](#).

Verifying the sample IT configuration

Run the sample stand-alone JMS application to send and receive messages through IBM MQ, and verify that you have configured IBM MQ correctly for use with the sample application.

Before you begin

Download the sample application package. Click the following link and save the file to the computer on which you are installing IBM MQ: [sampleJMSApp.zip](#), then extract the contents. The package contains a sample JMS application .jar file and batch files for running the application.

- The sample `sampleJMSApp.jar` file and the .cmd files must be in the same directory.
- The .cmd files use environment variables to set the class path for running the JMS application. When running the JMS application, if you see a `Java java.lang.NoClassDefFoundError`, you might need to adjust the class path line in the command file.

About this task

The JMS application comprises a requester client, which sends the initial message, and a responder client, which receives the message and sends a reply. The supplied batch files perform the following actions:

- `runresponder.cmd` opens a command prompt window in which the responder client starts then waits for a message.
- `runrequester.cmd` opens a separate command prompt window in which the requester client starts then sends a request message and receives a reply.

With two command prompt windows, you can see the actions of the requester and responder separately and more clearly.

Procedure

1. Double-click the `runresponder.cmd` file.

In the command prompt window, labeled **Responder window**, the responder client starts then waits for a message.

```
> Connection factory located in JNDI.> Destination located in JNDI.> Creating connection to
QueueManager.> Created connection.
> Waiting for message.
```

2. Double-click the `runrequester.cmd` file.

In the **Requester window**, observe the requester messages. In the **Responder window**, observe the updated responder messages; the message it received (from the requester client) and the reply message that it sent.

Results

In the command prompt window, labeled **Requester window**, the requester client shows the connection status, the message it sent, then the reply message that it received from the responder client:

```
> Connection factory located in JNDI.> Destination located in JNDI.> Creating connection to
QueueManager.> Connection created.
> Sending stock request for 'BakedBeans'> Sent Message
ID=ID:414d5120514d5f4c33344c3238482020c3cd094d20002b02
> Received Message ID=ID:414d5120514d5f4c33344c3238482020c3cd094d20002902 for 'B
akedBeans - 15 tins in stock'
> Closing connection to QueueManager.> Closed Connection.
-----
In this window, observe the messages sent through IBM MQ:
- The request message sent
- The reply message received
-----
When ready, press any key to close this window
Press any key to continue . . .
```

In the **Responder window**, observe the updated responder messages; the message it received (from the requester client) and the reply message that it sent:

```
> Connection factory located in JNDI.> Destination located in JNDI.> Creating connection to
QueueManager.> Created connection.
> Waiting for message.

> Received Message ID=ID:414d5120514d5f4c33344c3238482020c3cd094d20002b02 for 'B
akedBeans'
> Sending Reply Message 'BakedBeans - 15 tins in stock'> Sent Message
ID=ID:414d5120514d5f4c33344c3238482020c3cd094d20002902
> Closing connection to QueueManager.> Closed connection.
-----
In this window, observe the updated responder messages
- The request message received (from the requester)
- The reply message sent
-----
When ready, press any key to close this window
Press any key to continue . . .
```

The messages shown in the two command windows verify that the requester and responder clients of the sample application can communicate with each other through IBM MQ.

What to do next

You are now ready to start migrating your sample IBM MQ 9.3 installation to a later release of IBM MQ using one of the following two migration options:

- To migrate using the single-stage migration method, follow the instructions in [“Option 1: Single-stage migration”](#) on page 127.
- To migrate using the side-by-side migration method, follow the instructions in [“Option 2: Side-by-side migration”](#) on page 133.

Option 1: Single-stage migration

Option 1 of this scenario shows how to migrate from an earlier to a later version of IBM MQ when using the single-stage migration method. With a single-stage migration, the installation of a later version of IBM MQ replaces an earlier version in the same installation location.

Before you begin

The starting point for this scenario is the initial IT configuration described in [“Overview: Initial IT configuration”](#) on page 114.

Before starting this task, follow the instructions in [“Creating an initial IT configuration”](#) on page 118 to set up an initial IT configuration.

About this task

With a single-stage migration, you can choose either to uninstall the earlier version of IBM MQ before installing the newer version, or to install the newer version without first uninstalling the older version (that is, to migrate in place). In both cases, the later release is installed in the same directory as the earlier release. Option 1 of this scenario shows a single-stage migration where the older version is uninstalled before installing the newer version. The queue manager data is not removed as part of the uninstallation process, which means that the sample queue managers used in this scenario are retained and are detected when you install the newer version of IBM MQ.

Procedure

1. [Stop the queue managers](#) running in the earlier version of IBM MQ, and back up the queue manager data.
2. [Uninstall the earlier version of IBM MQ](#) that you are migrating from, without removing the queue manager data.
3. Install IBM MQ 9.4 using the launchpad..
4. Use IBM MQ Explorer [to verify the new IBM MQ 9.4 installation](#).
Verify that the queue managers have been successfully migrated from the earlier release and that you can put messages to and get messages from the migrated queues.

Related tasks

[Migrating on AIX and Linux: single-stage](#)

Preparing to migrate

Before migrating to a later version of IBM MQ, you must first stop the queue manager and back up the queue manager data.

About this task

If you migrate from a previous version of IBM MQ without first backing up your system, you cannot revert to the previous version, should you decide not to continue with the migration. Backing up your system before you install the new version enables you to back out the upgrade if necessary. If you do back out an upgrade, however, you cannot recover any work, such as changes to messages and objects, carried out by the later version of IBM MQ.

Before taking the backup, stop the queue manager that you are going to back up, which for this scenario is sampleQM. If you try to take a backup of a running queue manager, the backup might not be consistent because of any updates that were in progress when the files were copied.

Procedure

1. Open IBM MQ Explorer.
Click **Start > All Applications > IBM MQ > IBM MQ Explorer**.

2. Stop queue manager sampleQM.

- a) In the Navigator View, right-click the queue manager sampleQM.
- b) Click **Stop**.

The **End Queue Manager** window opens.

- c) Select **Controlled**, then click **OK**.

Selecting the Controlled option stops your queue manager in a controlled, orderly way. The **Immediate** option, which forces the queue manager to stop, is typically used only if a controlled stop fails to complete successfully.

The queue manager stops. In the IBM MQ, the icon next to queue manager sampleQM is changed to include a red arrow that points downward.

3. Close IBM MQ Explorer.

4. Back up the queue manager data.

Take copies of all of the following data, making sure that you include all backup directories. Some of the directories might be empty, but you need them all if you need to restore the backup at a later date, so save them too.

- The queue manager data located in C:\ProgramData\IBM\MQ\Qmgrs.
- The log file directories for the queue managers located in C:\ProgramData\IBM\MQ\log, including the log control file amqhlctl1.lfh.
- The configuration files located in the C:\ProgramData\IBM\MQ\Config.
- IBM MQ 9.3 .ini file and the registry entries. The queue manager information is stored in the .ini file and can be used to revert to a previous version of the product.

5. Stop IBM MQ.

- a) Stop the IBM MQ Service.

Right-click the **IBM MQ** icon in the system tray, then click **Stop IBM MQ**.

A dialog box with the following message is displayed:

Shutting down IBM MQ will terminate all running queue managers and IBM MQ processes. Are you sure you want to continue? (AMQ4102)

- b) Click **Yes**, then wait for IBM MQ to stop.
- c) When IBM MQ has stopped, right-click the **IBM MQ** icon in the system tray, then click **Exit**.

Results

You have stopped the queue manager that you are going to migrate to the later release of IBM MQ and have taken a backup of the queue manager data.

What to do next

You are now ready to uninstall IBM MQ as described in [“Uninstalling the earlier version” on page 128](#).

Related tasks

[Backing up queue manager data](#)

Uninstalling the earlier version

Uninstall the earlier version using the control panel. For a single-stage migration on Windows, uninstalling the earlier version of the product before installing the later version is optional.

Before you begin

Before starting this task, you must first stop the queue managers, close IBM MQ Explorer and stop IBM MQ as described in [“Preparing to migrate” on page 127](#).

About this task

In this task, you uninstall IBM MQ using the Windows control panel. The queue manager data is not removed as part of the uninstallation process, which means that the sample queue managers used in this scenario are retained and can be detected when you install the newer version of the product.

Whether or not you have to uninstall the earlier version of the product before you install the later version depends upon your operating system. On Windows systems, uninstalling is optional and you could alternatively install the newer version without uninstalling the earlier one. Note that in this case, some of the options and messages that you would see during the installation would be different from those that appear when you have uninstalled the earlier version first. For more information about the platforms on which you must uninstall the earlier version before installing the newer version, see [Migrating on AIX and Linux: single-stage](#).

Procedure

1. Open the Windows Control Panel by clicking **Start > Control Panel > Uninstall a program**.
2. In the **Programs and Features** window, find the entry for the installation that you want to remove, for example IBM WebSphere® MQ (Installation1), and click **Uninstall**.

The uninstallation process begins and runs to completion. When the process completes, the earlier version of IBM MQ is removed from your computer and is no longer displayed in the list of programs.

Results

The earlier version of IBM MQ has been removed from your computer. However the queue manager data has not been removed.

What to do next

You are now ready to install the later version of IBM MQ as described in [“Installing IBM MQ 9.4 using the launchpad”](#) on page 129.

Related tasks

[Uninstalling IBM MQ on Windows systems](#)

Installing IBM MQ 9.4 using the launchpad

Use the installation launchpad and wizards to install the later version of IBM MQ on the same Windows computer as the one on which the earlier version was installed.

Before you begin

Before starting this task, download the compressed file that contains the installation image, then uncompress it into a temporary directory.

This task assumes that you have previously uninstalled the earlier version of IBM MQ from which you are migrating as described in [“Uninstalling the earlier version”](#) on page 128. If you install the later version without first uninstalling the earlier version, some of the options and messages that you see during the installation process will be different from those described in this task.

Before starting this task, complete the following checks:

- You must have local administrator authority when you are installing. Define this authority through the Windows facilities.
- Ensure that the machine name does not contain any spaces.
- Ensure that you have sufficient disk space. For more information, see [Disk space requirements on Multiplatforms](#).

For this scenario, it is not necessary to determine whether you need to define Windows domain user ID's for any IBM MQ users, as this requirement is outside the scope of this scenario. See [Creating an Active Directory and DNS domain for IBM MQ](#) for more information.

Before you install IBM MQ, check that your system meets the hardware and software requirements. For the latest details of hardware and software requirements, see [System Requirements for IBM MQ](#).

About this task

This task describes the basic steps for installing IBM MQ on a Windows operating system when migrating from an earlier version.

Note: The default program and data directory locations are the same for IBM MQ 9.0 and later versions. Therefore you do not need to change the specification of the program and data directories when migrating from IBM MQ 9.0 to a later version. However, when migrating from versions before IBM MQ 9.0, there are differences in the default locations that you need to consider. For more information, see [Program and data directory locations on Windows](#).

The installation programs contain links to further information if you require it during the installation process. The installation process has the following parts:

1. Use the Launchpad to check and install software requirements, specify network information, and start the IBM MQ installation wizard.
2. Use the IBM MQ installation wizard to install the software, and to start the Prepare IBM MQ Wizard.
3. Use the Prepare IBM MQ Wizard to start the IBM MQ service.

Procedure

1. Start the installation process.

In Windows Explorer, navigate to the temporary folder into which you downloaded the installation image, then double-click `setup.exe`.

The installation launchpad is started.

2. Start the Launchpad, review, and if necessary, modify the software requirements and network configuration.

- a) Navigate to the IBM MQ software directory, and double-click the file `Setup.exe` to start the Launchpad.
- b) Click the **Software Requirements** button to display the **Software Requirements** tab.
- c) Check that the software requirements have been met and that the entry for the requirement displays a green tick with the words OK. Make any indicated corrections.

Note:

For details of any requirement, click the check box to expand an information tab.

- d) Click the **Network Configuration** button to display the **Network Configuration** tab.
- e) Click the **No** radio button.

Note: This scenario assumes that you do not need to configure a domain user ID for IBM MQ. For more information regarding configuring IBM MQ for Windows domain users, click the **More information** button.

- f) On the **IBM MQ Installation** tab of the Launchpad, select the installation language, and then click **Launch IBM MQ Installer** to start the IBM MQ installation wizard.

You have completed setting up IBM MQ by meeting or specifying your installation requirements, and have started the IBM MQ installation wizard.

3. Use the IBM MQ installation wizard to install the software, and to start the Prepare IBM MQ Wizard.
 - a) In the IBM MQ installation wizard, read the License Agreement and click the **I accept the terms in the license agreement** check box, and then click **Next**.

b) Click **Typical**, and then click **Next**.

c) On the **Ready to Install IBM MQ** page, review the installation information, then click **Install**.

The installation information includes the following details:

- Installation Name
- Top-level folder for Program Files
- Top level folder for Data Files

The following features are installed:

- IBM MQ Server
- IBM MQ: a graphical interface for administering and monitoring IBM MQ resources
- Java™ and .NET Messaging and Web Services
- IBM MQ Development Toolkit

The installation process begins. Depending on your system the installation process can take several minutes.

At the end of the installation process, the IBM MQ Setup window displays the message **Installation Wizard Completed Successfully**.

d) Click **Finish**.

You have successfully installed IBM MQ. The Prepare IBM MQ Wizard starts automatically, displaying the **Welcome to the Prepare IBM MQ Wizard** page.

4. Use the Prepare IBM MQ Wizard to start the IBM MQ service.

a) On the Welcome to the Prepare IBM MQ Wizard page, select **Next**.

The Prepare IBM MQ Wizard displays the message **Status: Checking IBM MQ Configuration** and a progress bar. When the process is complete the IBM MQ Network Configuration page is displayed.

b) On the IBM MQ Network Configuration page of the Prepare IBM MQ Wizard, select **No**.

c) Click **Next**.

The Prepare IBM MQ Wizard displays a message **Status: starting the IBM MQ Service**, and a progress bar. When the process is complete the wizard displays the **Completing the Prepare IBM MQ Wizard** page.

d) Select **Launch IBM MQ Explorer** and choose whether to start Notepad to view the release notes, then click the **Finish** button.

IBM MQ Explorer starts.

Results

You have installed IBM MQ, and you have started IBM MQ Explorer.

What to do next

Now that you have installed the later version of IBM MQ, you are ready to check that the sample queue managers have been successfully migrated and that you are able to put messages to and get messages from the migrated queues as described in [“Verifying your IBM MQ 9.4 installation” on page 132](#).

Related concepts

[Where to find downloadable installation images](#)

Related tasks

[Installing an IBM MQ server on Windows](#)

Verifying your IBM MQ 9.4 installation

After you install IBM MQ 9.4, use IBM MQ Explorer to verify that the queue managers and queues have successfully migrated from the earlier release then verify that you can use the sample application.

About this task

When you have checked that the migrated queue manager, sampleQM, is visible in the Navigator view of IBM MQ Explorer, verify that you can put a message to and get a message from the migrated queue, then check that you can still run the sample application.

Procedure

1. If IBM MQ Explorer is not running, start it now.
Click **Start > All Programs > IBM MQ > IBM MQ Explorer**.
2. Verify that your queue managers have been successfully migrated to the later version of IBM MQ:
 - a) In the Navigator view, expand the **Queue Managers** folder.
 - b) Check that you can see the queue manager sampleQM in the **Queue Managers** folder.
 - c) Expand queue manager sampleQM, click the **Queues** folder, and check that you can see queue Q1 in the Content view.
3. If queue manager sampleQM is not already started, start it now.
 - a) In the Navigator view, expand the queue managers node.
 - b) Right-click queue manager sampleQM, then click **Start**.
4. Verify that you can put a message to queue Q1.
 - a) In the Navigator view, expand the **Queue Managers** folder.
 - b) Expand queue manager sampleQM, then click the **Queues** folder.
 - c) In the Content view, right-click queue Q1, then click **Put Test Message**.
The **Put test message** dialog opens.
 - d) In the **Message data** field, type some text, for example Hello queue!, then click **Put message**.
The **Message data** field is cleared and the message is put on the queue.
 - e) Click **Close**.
In the Content view, notice that the **Current queue depth** value of the queue is now 1. If the Current queue depth column is not visible, you might need to scroll to the right of the Content view.
5. Verify that you can get the message from queue Q1.
 - a) In the Navigator view, expand the **Queue Managers** folder,
 - b) Expand queue manager sampleQM then click the **Queues** folder.
 - c) In the Content view, right-click queue Q1, then click **Browse Messages**.
The Message browser opens to show the list of the messages that are currently on the queue.
 - d) Double-click the last message to open the properties dialog.
On the **Data** page of the properties dialog, the **Message data** field displays the content of the message in human-readable form.
6. Verify that you can run the sample application.
 - a) Double-click the `runresponder.cmd` file.
In the command prompt window, labeled **Responder window**, the responder client starts then waits for a message.

```
> Connection factory located in JNDI.> Destination located in JNDI.> Creating connection to QueueManager.> Created connection.
> Waiting for message.
```
 - b) Double-click the `runrequester.cmd` file.

In the **Requester window**, observe the requester messages. In the **Responder window**, observe the updated responder messages; the message it received (from the requester client) and the reply message that it sent.

In the command prompt window, labeled **Requester window**, the requester client shows the connection status, the message it sent, then the reply message that it received from the responder client:

```
> Connection factory located in JNDI.> Destination located in JNDI.> Creating connection to
QueueManager.> Connection created.
> Sending stock request for 'BakedBeans'> Sent Message
ID=ID:414d5120514d5f4c33344c3238482020c3cd094d20002b02
> Received Message ID=ID:414d5120514d5f4c33344c3238482020c3cd094d20002902 for 'B
akedBeans - 15 tins in stock'
> Closing connection to QueueManager.> Closed Connection.
```

In this window, observe the messages sent through IBM MQ:

- The request message sent
- The reply message received

When ready, press any key to close this window
Press any key to continue . . .

In the **Responder window**, observe the updated responder messages; the message it received (from the requester client) and the reply message that it sent:

```
> Connection factory located in JNDI.> Destination located in JNDI.> Creating connection to
QueueManager.> Created connection.
> Waiting for message.
```

```
> Received Message ID=ID:414d5120514d5f4c33344c3238482020c3cd094d20002b02 for 'B
akedBeans'
> Sending Reply Message 'BakedBeans - 15 tins in stock'> Sent Message
ID=ID:414d5120514d5f4c33344c3238482020c3cd094d20002902
> Closing connection to QueueManager.> Closed connection.
```

In this window, observe the updated responder messages

- The request message received (from the requester)
- The reply message sent

When ready, press any key to close this window
Press any key to continue . . .

The messages shown in the two command windows verify that the requester and responder clients of the sample application can communicate with each other through IBM MQ.

Results

You have successfully migrated to the later version of IBM MQ.

Option 2: Side-by-side migration

Option 2 of this scenario shows how to migrate from an earlier to a later release of IBM MQ when using the side-by-side migration method. With a side-by-side migration, you install the latest version of IBM MQ alongside the earlier version that you want to migrate from. Queue managers and applications continue to be associated with the earlier release until you migrate them to the later release.

Before you begin

The starting point for this scenario is the initial IT configuration described in [“Overview: Initial IT configuration”](#) on page 114.

Before starting this task, follow the instructions in [“Creating an initial IT configuration”](#) on page 118 to set up an initial IT configuration.

About this task

When you follow the side-by-side migration method described in this scenario, you install the later version alongside the earlier version in an alternate location. Since you uninstall the earlier version before starting any queue managers in the later version, you can assign your installation of the later version of

IBM MQ to be the primary installation. For more information about the primary installation, see [Primary installation](#).

Procedure

1. [Install IBM MQ 9.4 using the launchpad and then verify the installation.](#)
2. [Stop the queue managers running in the earlier version of IBM MQ.](#)
3. [Uninstall the earlier version of IBM MQ.](#)
4. [Make IBM MQ 9.4 the primary installation.](#)
5. Optional: [Associate the queue managers with IBM MQ 9.4](#)
6. Use IBM MQ Explorer to [verify the IBM MQ 9.4 installation](#).

Check that the queue managers have been successfully migrated from the earlier release and that you can put messages to and get messages from the migrated queues.

Related tasks

[Migrating on Windows: side-by-side](#)

Installing IBM MQ 9.4 using the launchpad

Use the installation launchpad and wizards to install the later version of IBM MQ alongside the earlier version on Windows.

Before you begin

Before completing this task, complete the following checks:

- You must have local administrator authority when you are installing. Define this authority through the Windows facilities.
- Ensure that the machine name does not contain any spaces.
- Ensure that you have sufficient disk space, up to 1005 MB, to fully install IBM MQ for Windows.
- Determine whether you need to define Windows domain user ID's for any IBM MQ users.

Before you install IBM MQ, check that your system meets the hardware and software requirements. For the latest details of hardware and software requirements on all supported platforms, see [System Requirements for IBM MQ](#).

About this task

This task describes the basic steps for installing IBM MQ on Windows if you do not already have it installed on your system.

This task assumes that you will be using the default IBM MQ program and data files locations.

Note: The default program and data directory locations are the same for IBM MQ 9.0 and later versions. Therefore you do not need to change the specification of the program and data directories when migrating from IBM MQ 9.0 to a later version. However, when migrating from versions before IBM MQ 9.0, there are differences in the default locations that you need to consider. For more information, see [Program and data directory locations on Windows](#).

The installation programs contain links to further information if you require it during the installation process. The installation process has the following parts:

1. Use the Launchpad to check and install software requirements, specify network information, and start the IBM MQ installation wizard.
2. Use the IBM MQ installation wizard to install the software, and to start the Prepare IBM MQ Wizard.
3. Use the Prepare IBM MQ Wizard to start the IBM MQ service.

Procedure

1. Start the installation process.

In Windows Explorer, navigate to the temporary folder into which you downloaded the installation image, then double-click `setup.exe`.

The installation launchpad is started.

2. Using the launchpad, review and, if necessary, modify the software requirements and network configuration.

- a) Click the **Software Requirements** button to display the **Software Requirements** tab.
- b) Check that the software requirements have been met and that the entry for the requirement displays a green tick with the words OK. Make any indicated corrections.

Note: To see more details about any requirement, click the plus (+) button.

- c) Click the **Network Configuration** button to display the **Network Configuration** tab.
- d) Click the **No** radio button.

Note: This scenario assumes that you do not need to configure a domain user ID for IBM MQ. For more information regarding configuring IBM MQ for Windows domain users, click the **More information** button.

- e) On the **IBM MQ Installation** tab of the Launchpad, select the installation language, and then click the **Launch IBM MQ Installer** button to start the IBM MQ installation wizard.

You have completed your review IBM MQ of the installation requirements, have made any required modifications, and have started the IBM MQ installation wizard.

3. Use the IBM MQ installation wizard to install the software, and to start the Prepare IBM MQ Wizard.

The IBM MQ installation wizard checks for existing installations and displays the upgrade or installation options that are available to you. In the case of this scenarios, there are two options:

- Install leaving the existing installation(s) untouched
 - Upgrade 8.0.0.5 installation 'Installation 1'
- a) Select **Install leaving the existing installation(s) untouched**, and then click **Next**.
 - b) Read the License Agreement and click the **I accept the terms in the license agreement** check box, and then click **Next**.
 - c) Click **Typical**, and then click **Next**.
 - d) On the **Ready to Install IBM MQ** page, review the displayed installation information and click **Install**.

The installation information includes the following details:

- Installation Name
- Top-level folder for Program Files
- Top level folder for Data Files

The following features are installed:

- IBM MQ Server
- IBM MQ: a graphical interface for administering and monitoring IBM MQ resources
- Java™ and .NET Messaging and Web Services
- IBM MQ Development Toolkit

The installation process begins. Depending on your system the installation process can take several minutes.

At the end of the installation process, the IBM MQ Setup window displays the message **Installation Wizard Completed Successfully**.

- e) Click **Finish**.

You have successfully installed IBM MQ. The Prepare IBM MQ Wizard starts automatically, displaying the **Welcome to the Prepare IBM MQ Wizard** page.

4. Use the Prepare IBM MQ Wizard to start the IBM MQ service.

a) On the Welcome to the Prepare IBM MQ Wizard page, select **Next**.

The Prepare IBM MQ Wizard displays the message Status: Checking IBM MQ Configuration and a progress bar. When the process is complete the IBM MQ Network Configuration page is displayed.

b) On the IBM MQ Network Configuration page of the Prepare IBM MQ Wizard, select **No**.

c) Click **Next**.

The Prepare IBM MQ Wizard displays a message Status: starting the IBM MQ Service, and a progress bar. When the process is complete the wizard displays the Completing the Prepare IBM MQ Wizard page.

d) Select **Launch IBM MQ Explorer** and choose whether to start Notepad to view the release notes, then click the **Finish** button.

IBM MQ Explorer starts.

What to do next

You have installed the later version of IBM MQ alongside the earlier version, but in a different installation directory, and you have started IBM MQ Explorer.

You are now ready to stop the queue managers running in the earlier version of IBM MQ as described in [“Stopping the queue manager” on page 136](#).

Stopping the queue manager

Before migrating to a later version of IBM MQ, you must first stop the queue manager and back up the queue manager data.

About this task

Before taking the backup, stop the queue manager that you are going to back up. If you try to take a backup of a running queue manager, the backup might not be consistent because of any updates that were in progress when the files were copied.

Procedure

1. Open IBM MQ Explorer.

Click **Start > All Applications > IBM MQ > IBM MQ Explorer**.

2. Stop queue manager sampleQM.

a) In the Navigator View, right-click the queue manager sampleQM.

b) Click **Stop**.

The **End Queue Manager** window opens.

c) Select **Controlled**, then click **OK**.

Selecting the **Controlled** option stops your queue manager in a controlled, orderly way. The **Immediate** option, which forces the queue manager to stop, is typically used only if a controlled stop fails to complete successfully.

The queue manager stops. In the IBM MQ, the icon next to queue manager sampleQM is changed to include a red arrow that points downward.

3. Close IBM MQ Explorer.

4. Back up the queue manager data.

Take copies of all of the following data, making sure that you include all backup directories. Some of the directories might be empty, but you need them all if you need to restore the backup at a later date, so save them too.

- The queue manager data located in C:\ProgramData\IBM\MQ\Qmgrs.
- The log file directories for the queue managers located in C:\ProgramData\IBM\MQ\log, including the log control file amqh1ctl1.lfh.
- The configuration files located in the C:\ProgramData\IBM\MQ\Config.
- IBM MQ 9.3 .ini file and the registry entries. The queue manager information is stored in the .ini file and can be used to revert to a previous version of the product.

5. Stop IBM MQ.

- a) Stop the IBM MQ Service.

Right-click the **IBM MQ** icon in the system tray, then click **Stop IBM MQ**.

A dialog box with the following message is displayed:

Shutting down IBM MQ will terminate all running queue managers and IBM MQ processes. Are you sure you want to continue? (AMQ4102)

- b) Click **Yes**, then wait for IBM MQ to stop.
- c) When IBM MQ has stopped, right-click the **IBM MQ** icon in the system tray, then click **Exit**.

What to do next

After stopping the queue managers, you are ready to associate them with your new installation of the later version of IBM MQ as described in [“Associating the queue managers with IBM MQ 9.4”](#) on page 139.

Uninstalling the earlier version

Uninstall the earlier version of the product using the Windows control panel.

Before you begin

Before starting this task, you must first stop the queue managers, close IBM MQ Explorer and stop IBM MQ as described in [“Stopping the queue manager”](#) on page 136.

About this task

In this task, you uninstall IBM MQ using the Windows control panel. The queue manager data is not removed as part of the uninstallation process, which means that the sample queue managers used in this scenario are retained and can be detected when you install the newer version of the product.

Procedure

1. Open the Windows Control Panel by clicking **Start > Control Panel > Uninstall a program**.
2. In the **Programs and Features** window, find the entry for the installation that you want to remove, for example IBM WebSphere MQ (Installation1), and click **Uninstall**.

The uninstallation process begins and runs to completion. When the process completes, the earlier version of IBM MQ is removed from your computer and is no longer displayed in the list of programs.

Results

The earlier version of the product has been removed from your computer. However the queue manager data has not been removed.

What to do next

You are now ready to make the later version of IBM MQ the primary installation as described in [“Making IBM MQ 9.4 the primary installation”](#) on page 138.

Related tasks

[Uninstalling IBM MQ on Windows systems](#)

Making IBM MQ 9.4 the primary installation

Before you start any queue managers in your new installation of the later version of IBM MQ, you can, optionally, make the later version the primary installation.

About this task

On systems that support multiple installations of IBM MQ, the primary installation is the one to which IBM MQ system-wide locations refer. Having a primary installation is optional, but convenient.

When you follow the side-by-side migration method described in this scenario, because you uninstall the earlier version before starting any queue managers in the later release, you can assign your installation of the later version of the product to be the primary installation.

For more information about the primary installation, see [Primary installation](#).

Procedure

1. Check the current primary installation by entering the **dspmqinst** command into the command prompt.

The command prompt displays the details of any current installations. The current primary installation has the following line **Primary: Yes**.

2. Use the **setmqinst** command to change the current primary installation.

In the command prompt, enter:

```
setmqinst -x -n Installation_Name
```

where *Installation_Name* is the name of the current primary installation.

If the command is successful, the command prompt displays the message '*Installation_Name*' (*Filepath*) has been unset as the Primary Installation.

3. Use the **setmqinst** command to set the new IBM MQ 9.4 installation as the primary installation.

In the command prompt, enter:

```
setmqinst -i -n V9_Installation
```

where *V9_Installation* is the name of the IBM MQ 9.4 installation.

If the command is successful, the command prompt displays the message '*V9_Installation*' (*Filepath*) has been set as the primary installation. You must restart the operating system to complete the update.

Note: As directed in the success message, you must restart the operating system to complete the update.

What to do next

You are ready to associate the migrated queue managers with the later version of IBM MQ as described in [“Associating the queue managers with IBM MQ 9.4” on page 139](#).

Use the Transfer queue managers wizard to associate the sampleQM queue manager with your installation of the later version of IBM MQ.

Before you begin

Before starting this task, make sure that you have stopped the queue manager as described in [“Stopping the queue manager”](#) on page 136, otherwise you will not be able to complete the transfer.

About this task

The Transfer queue managers wizard feature of IBM MQ Explorer enables you to transfer one or more queue managers from other installations to the current installation. This wizard is equivalent to the **setmqm** command, but saves having to type the required paths and parameters. Only stopped queue managers can be transferred; running queue managers are shown for reference.

Once you have transferred and started a queue manager on an installation for a later version of the product, it is not possible to migrate back to an earlier version.

Procedure

1. Start IBM MQ Explorer.

Click **Start > All Applications > IBM MQ > IBM MQ Explorer**.

2. In the Navigator view, right-click the queue managers node and select **Transfer Queue Managers**.
3. Right-click then select the queue manager sampleQM, then click **Transfer**.

The **setmqm** command is invoked with the selected queue managers. If the transfer is successful, the navigator tree updates to include the transferred queue managers. If there are any problems, a dialog is shown with the error message from the command.

4. Start the queue manager sampleQM.
 - a) In the Navigator view, expand the queue managers node.
 - b) Right-click the name of the queue manager, then click **Start**.

Results

You have successfully associated the queue manager sampleQM with the later version of IBM MQ.

What to do next

Verify that the queue manager sampleQM has been successfully migrated by confirming that you can put a message to and get a message from the queue as described in [“Verifying the IBM MQ 9.4 installation”](#) on page 139.

Verifying the IBM MQ 9.4 installation

After installing the later version of IBM MQ, use IBM MQ Explorer to verify that the queue managers and queues have successfully migrated from the earlier release then verify that you can use the sample application.

About this task

When you have checked that the migrated queue manager, sampleQM, is visible in the Navigator view of IBM MQ Explorer, verify that you can put a message to and get a message from the migrated queue, then check that you can still run the sample application.

Procedure

1. If IBM MQ Explorer is not running, start it now.

Click **Start > All Programs > IBM MQ > IBM MQ Explorer**.

2. Verify that your queue managers have been successfully migrated to the later version of IBM MQ:
 - a) In the Navigator view, expand the **Queue Managers** folder.
 - b) Check that you can see the queue manager sampleQM in the **Queue Managers** folder.
 - c) Expand queue manager sampleQM, click the **Queues** folder, and check that you can see queue Q1 in the Content view.
3. If queue manager sampleQM is not already started, start it now.
 - a) In the Navigator view, expand the queue managers node.
 - b) Right-click queue manager sampleQM, then click **Start**.
4. Verify that you can put a message to queue Q1.
 - a) In the Navigator view, expand the **Queue Managers** folder.
 - b) Expand queue manager sampleQM, then click the **Queues** folder.
 - c) In the Content view, right-click queue Q1, then click **Put Test Message**.
The **Put test message** dialog opens.
 - d) In the **Message data** field, type some text, for example Hello queue!, then click **Put message**.
The **Message data** field is cleared and the message is put on the queue.
 - e) Click **Close**.
In the Content view, notice that the **Current queue depth** value of the queue is now 1. If the Current queue depth column is not visible, you might need to scroll to the right of the Content view.
5. Verify that you can get the message from queue Q1.
 - a) In the Navigator view, expand the **Queue Managers** folder,
 - b) Expand queue manager sampleQM then click the **Queues** folder.
 - c) In the Content view, right-click queue Q1, then click **Browse Messages**.
The Message browser opens to show the list of the messages that are currently on the queue.
 - d) Double-click the last message to open the properties dialog.
On the **Data** page of the properties dialog, the **Message data** field displays the content of the message in human-readable form.
6. Verify that you can run the sample application.
 - a) Double-click the `runresponder.cmd` file.
In the command prompt window, labeled **Responder window**, the responder client starts then waits for a message.

```
> Connection factory located in JNDI.> Destination located in JNDI.> Creating connection to QueueManager.> Created connection.
> Waiting for message.
```
 - b) Double-click the `runrequester.cmd` file.
In the **Requester window**, observe the requester messages. In the **Responder window**, observe the updated responder messages; the message it received (from the requester client) and the reply message that it sent.

In the command prompt window, labeled **Requester window**, the requester client shows the connection status, the message it sent, then the reply message that it received from the responder client:

```
> Connection factory located in JNDI.> Destination located in JNDI.> Creating connection to QueueManager.> Connection created.
> Sending stock request for 'BakedBeans'> Sent Message
ID=ID:414d5120514d5f4c33344c3238482020c3cd094d20002b02
> Received Message ID=ID:414d5120514d5f4c33344c3238482020c3cd094d20002902 for 'BakedBeans - 15 tins in stock'
> Closing connection to QueueManager.> Closed Connection.
```

In this window, observe the messages sent through IBM MQ:

- The request message sent
 - The reply message received
-

When ready, press any key to close this window
Press any key to continue . . .

In the **Responder window**, observe the updated responder messages; the message it received (from the requester client) and the reply message that it sent:

```
> Connection factory located in JNDI.> Destination located in JNDI.> Creating connection to
QueueManager.> Created connection.
> Waiting for message.
```

```
> Received Message ID=ID:414d5120514d5f4c33344c3238482020c3cd094d20002b02 for 'B
akedBeans'
> Sending Reply Message 'BakedBeans - 15 tins in stock'> Sent Message
ID=ID:414d5120514d5f4c33344c3238482020c3cd094d20002902
> Closing connection to QueueManager.> Closed connection.
```

```
-----
In this window, observe the updated responder messages
- The request message received (from the requester)
- The reply message sent
-----
```

When ready, press any key to close this window
Press any key to continue . . .

The messages shown in the two command windows verify that the requester and responder clients of the sample application can communicate with each other through IBM MQ.

Results

You have successfully migrated to the later version of IBM MQ.

What to do next

For more information about migrating and upgrading see [Maintaining and migrating](#).

Installing a later version of IBM MQ to coexist with an earlier version on Windows

This scenario shows all the steps to install a Long Term Support (LTS) version of IBM MQ, while co-existing ("side-by-side") with an earlier version of the product. Furthermore, the steps include the installation of a Fix Pack to the later version. You should always install the latest Fix Pack level.

Overview of multiple installations

A description of multiple installations together with the hardware and software used in this scenario.

About this task


An important aspect of this feature of multiple installations of IBM MQ (multi-install) in the same host, is that this type of installation does not require that the queue managers and applications on the earlier version of the product do not need to be stopped when doing activities with the later version of the product.

That is, the installation of the later version of the product does not affect the running of applications on the earlier version of the product. This is helpful when you try to do a multi-stage migration of the queue managers on the earlier version of the product to the later version.

For this scenario, IBM MQ 8.0.0 is used as the earlier version of the product, and IBM MQ 9.1.0 is used as the later version of the product.

Using a Long Term Support version of IBM MQ

This scenario uses an LTS version of IBM MQ for the later version of the product.

 If you are using a Continuous Delivery (CD) version of IBM MQ for the later version of the product, you must uninstall the CD version you are using, for example, IBM MQ 9.1.1, before you install

a later version, for example IBM MQ 9.1.5. For more information, see [Migrating from one Continuous Delivery release to another](#).

Hardware and software used for this scenario

Operating System

Windows 10

Hostname: johndoe1.fyre.<yourdomainname>.com

Queue managers

QM80

Created with IBM MQ 8.0.0; to remain at IBM MQ 8.0.0

QMMIG

Created with IBM MQ 8.0.0; to be migrated to IBM MQ 9.1.0

QM910

Created with IBM MQ 9.1.0; to remain at IBM MQ 9.1.0

Installing a later version of IBM MQ side-by-side to an earlier version.

How you install IBM MQ 9.4 side-by-side to the existing version of the product on the same machine. The IBM MQ 9.1 installation will not be designated as the primary installation.

Before you begin

Ensure IBM MQ 9.1 is installed on your system. Use the following instructions, [Installing the IBM MQ server on Windows](#), to install the product if IBM MQ 9.1 is not installed.

You need to have selected the *Custom* option so that you can explicitly select the Client. See [Installation methods for Windows](#) for more information.

Also you need to have invoked the [setmqenv](#) command with the **-n** parameter to have set the installation name to `Installation1`. It is helpful if you use a batch file to invoke the command.

About this task

All the queue manager data is stored in a common directory structure. Even though the executable code for each version of IBM MQ is stored in a different directory structure, the data for all the queue managers, regardless of the version, is stored in: `MQ_DATA_PATH=C:\ProgramData\IBM\MQ`.

To install IBM MQ 9.4:

Procedure

1. Login as an administrator.

IBM MQ 9.4 will be installed in the default directory `C:\Program Files\IBM\MQ`.

2. Go to the directory where the download file is located, for example, `C:\downloads\mq9300`.
3. Unzip the downloaded file.

The files are extracted into a new subdirectory called `MQServer1`.

4. Switch to the new directory and issue the command **setup.exe** to start the installer.

- a) Click on *Software Requirements* to check that your enterprise has the requisite software installed. See [Checking requirements on Windows](#) for more information. In this scenario, the system has the requisite requirements.

- b) Click on *Network Configuration*.

In this scenario, the machine is not part of a Domain, so there is no need to indicate a domain user and the answer to the question is No.

- c) Click on *IBM MQ Installation*.

- d) Click on *Launch IBM MQ Installer*.

The installer detects that there are other installations on the system and displays the message:

Upgrade or install

Upgrade the existing installation or install the new version alongside it

For this scenario, leave the other installations intact, so select the first entry *Install leaving the existing installation(s) untouched*.

- e) Click **Next** and accept the license.

- f) Select *Custom* for the installation option.

See [Installation methods for Windows](#) for more information.

- g) Click **Next**.

In the

Installation Details

Define the installation details

you see:

Installation Name

Installation2

Installation folder for program files

C:\ProgramFiles\IBM\MQ

The list of features to be installed is shown. Note that some items:

- Are not installed by default
- Need to be selected if you want to install them.

For this scenario select *MQI client*, and you should select this option anyway.

- h) Click **Next**.

In the screen entitled *Ready to Install IBM MQ* you see the summary of the Installation Name, location, components to install, and so on.

- i) Click **Install** to proceed.

Copying of the files into the installation directory structure begins. After copying the files, the following dialog appears: The IBM MQ Installation Wizard has successfully installed IBM MQ.

- j) Click **Finish**.

After the installation, you see the following dialog: Welcome to the Prepare IBM MQ Wizard. Because this scenario is not using a Windows Domain, accept the default of **No**.

Accept to launch IBM MQ Explorer, and unless you have a very specific reason, you can accept the default setting of *Start MQ Explorer with a new workspace*.

Results

You have successfully installed another version of IBM MQ for Windows alongside an existing version of the product.

What to do next

You need to run the **setmqenv** command to use the commands on either version. See [“Using the setmqenv command to run with both versions of IBM MQ”](#) on page 144 for details.

Using the `setmqenv` command to run with both versions of IBM MQ

The installation activities are complete and you can now verify the directories containing the IBM MQ code.

Before you begin

Ensure that you have correctly installed the IBM MQ 8.0.0 product, Installation1, in C:\Program Files\IBM\WebSphere MQ\ and the IBM MQ 9.1 product, Installation2, in C:\Program Files\IBM\MQ.

About this task

Use the `dspmqiinst` command to display the installation information of the versions installed in the system without the need to take a look at the directories, and the `dspmqver` command to display the version information.

On Windows systems the information is kept in the registry and the installation configuration information is held in the following key: Computer\HKEY_LOCAL_MACHINE\SOFTWARE\IBM\WebSphere MQ\Installation.

Important: You must not edit or reference this key directly.

Procedure

1. Display the installation information using the `dspmqiinst` command.

```
InstName:      Installation1
InstDesc:
Identifier:    1
InstPath:      C:\Program Files\IBM\WebSphere MQ
Version:       8.0.0.9
Primary:       Yes
State:         Available
MSIProdCode:   {74F6B169-7CE6-4EFB-8A03-2AA7B2DBB57C}
MSIMedia:      8.0 Server
MSIInstanceId: 1

InstName:      Installation2
InstDesc:
Identifier:    2
InstPath:      C:\Program Files\IBM\MQ
Version:       9.1.0.0
Primary:       No
State:         Available
MSIProdCode:   {5D3ECA81-BF8D-4E80-B36C-CBB1D69BC110}
MSIMedia:      9.1 Server
MSIInstanceId: 1
```

Note: The respective installation names (InstName) are important.

2. Display the version of IBM MQ using `dspmqver` for the default (or primary) installation:

```
C:\> dspmqver
Name:      WebSphere MQ
Version:    8.0.0.9
Level:      p800-009-180321.1
BuildType:  IKAP - (Production)
Platform:   WebSphere MQ for Windows (x64 platform)
Mode:       64-bit
O/S:        Windows 10 Professional x64 Edition, Build 18363
InstName:   Installation1
InstDesc:
Primary:    Yes
InstPath:   C:\Program Files\IBM\WebSphere MQ
DataPath:   C:\ProgramData\IBM\MQ
MaxCmdLevel: 802
LicenseType: Production
```


After running the command you receive the following message: Note there are a number (1) of other installations, use the '-i' parameter to display them.

3. Issue the command, `C:\> where dspmqver`, and you see information about the primary installation:

```
C:\> where dspmqver
C:\Program Files\IBM\WebSphere MQ\bin64\dspmqver.exe
C:\Program Files\IBM\WebSphere MQ\bin\dspmqver.exe
```

4. To see the information about the IBM MQ 9.1 product issue the following command, `C:\> setmqenv -n Installation2`.
5. Issue the command `C:\> where dspmqver` again and you see information about the second installation:

```
C:\> where dspmqver
C:\Program Files\IBM\MQ\bin64\dspmqver.exe
C:\Program Files\IBM\MQ\bin\dspmqver.exe
```

6. Issue the command `C:\> dspmqver` again.

You now see:

```
C:\> dspmqver
Name:          IBM MQ
Version:       9.1.0.0
Level:         p910-L180705
BuildType:     IKAP - (Production)
Platform:      IBM MQ for Windows (x64 platform)
Mode:          64-bit
O/S:           Windows 10 Professional x64 Edition, Build 18363
InstName:      Installation2
InstDesc:
Primary:       No
InstPath:      C:\Program Files\IBM\MQ
DataPath:      C:\ProgramData\IBM\MQ
MaxCmdLevel:   910
LicenseType:   Production
```

7. Issue the command `C:\> set MQ`, and after using **setmqenv** you see information about the second installation.

```
C:\> set MQ
MQ_DATA_PATH=C:\ProgramData\IBM\MQ
MQ_ENV_MODE=64
MQ_FILE_PATH=C:\Program Files\IBM\MQ
MQ_INSTALLATION_NAME=Installation2
MQ_INSTALLATION_PATH=C:\Program Files\IBM\MQ
MQ_JAVA_DATA_PATH=C:\ProgramData\IBM\MQ
MQ_JAVA_INSTALL_PATH=C:\Program Files\IBM\MQ\java
MQ_JAVA_LIB_PATH=C:\Program Files\IBM\MQ\java\lib64
MQ_JRE_PATH=C:\Program Files\IBM\MQ\java\jre
```

You can create a batch file that will run the **setmqenv** command with the specified syntax. Ensure you have this batch file in a directory in your PATH, for example, `C:\WinTools`.

For example, you can create the batch file `set-mq-910.bat` with the contents:

```
REM Setup the environment to run MQ 9.1
CALL "C:\Program Files\IBM\MQ\bin\setmqenv" -n Installation2
REM Adding Samples to the path
SET PATH=%PATH%;%MQ_FILE_PATH%\Tools\c\Samples\Bin;%MQ_FILE_PATH%\Tools\c\Samples\Bin64
;%MQ_FILE_PATH%\Tools\jms\samples;%MQ_JAVA_INSTALL_PATH%\bin\ dspmqver -f 2
```

Notes:

- a. You need to use the "CALL" argument when invoking **setmqenv**. Without this argument, the processing of **setmqenv** ends the batch and does not allow following statements to run. That is, with the CALL argument, you allow other statements in the batch file to be processed.
- b. If you add an IBM MQ directory into your PATH, such as the location for the C-samples: `PATH=...;C:\Program Files\IBM\MQ\tools\c\Samples\Bin;...` this directory will be REMOVED by **setmqenv** the next time the command runs.

If you want to be able to run the C-samples, the last line in the preceding batch file is needed, in order to place the directory for the samples back into the PATH.

Notice also that MQ_FILE_PATH is used in order to use the proper directory structure for IBM MQ 9.1: SET PATH=%PATH%;%MQ_FILE_PATH%\tools\c\Samples\Bin.

Creating a queue manager

How you create a queue manager, using the **crtmqm** command. You can use IBM MQ Explorer to perform this task.

Before you begin

Use the **dspmq** command with the **-o installation** and **-s** parameters to show the installation name and status of the current queue managers.

```
C:\> dspmq -o installation -s
QMNAME(QM80) STATUS(Running) INSTNAME(Installation1)
INSTPATH(C:\Program Files\IBM\WebSphere MQ) INSTVER(9.1.0.9)
QMNAME(QMMIG) STATUS(Running) INSTNAME(Installation1)
INSTPATH(C:\Program Files\IBM\WebSphere MQ) INSTVER(9.1.0.9)
```

About this task

You must open a Windows command prompt and set yourself up as an *Administrator* to carry out the following process. If you try and issue the **crtmqm** command without using the command prompt, you receive message AMQ7077: You are not authorized to perform the requested operation.

Procedure

1. Select **Start > Windows System > Command Prompt > More > Run as administrator**

The title for the window created is *Administrator: Command Prompt*.

Note: The version of IBM MQ in the command prompt is IBM MQ 9.1.

2. Run the **setmqenv** command or the batch file you might have created, **set-mq-930**.

See [“Using the setmqenv command to run with both versions of IBM MQ” on page 144](#) for details.

In either case, you see Version 9.3.0.0

3. Issue the following command, C:\> **crtmqm -u SYSTEM.DEAD.LETTER.QUEUE QM930**

You see the following output:

```
IBM MQ queue manager created.
Directory 'C:\ProgramData\IBM\MQ\qmgrs\QM930' created.
The queue manager is associated with installation 'Installation2'.
Creating or replacing default objects for queue manager 'QM930'.
Default objects statistics : 87 created. 0 replaced. 0 failed.
Completing setup.
Setup completed.
```

4. Issue the following command to start the queue manager C:\> **strmqm QM930**

You see the following output. Note the lines that indicate the installation and the version under which the queue manager is running:

```
IBM MQ queue manager 'QM930' starting.
The queue manager is associated with installation 'Installation2'.
5 log records accessed on queue manager 'QM930' during the log replay phase.
Log replay for queue manager 'QM930' complete.
Transaction manager state recovered for queue manager 'QM930'.
IBM MQ queue manager 'QM930' started using V9.3.0.0.
```

5. Issue the command C:\> **dspmq -o installation -s** again to display the installed queue managers:

```
C:\> dspmq -o installation -s
QMNAME(QM80) STATUS(Running) INSTNAME(Installation1)
INSTPATH(C:\Program Files\IBM\WebSphere MQ) INSTVER(9.1.0.9)
QMNAME(QMMIG) STATUS(Running) INSTNAME(Installation1)
INSTPATH(C:\Program Files\IBM\WebSphere MQ) INSTVER(9.1.0.9)
QMNAME(QM910) STATUS(Running) INSTNAME(Installation2)
INSTPATH(C:\Program Files\IBM\MQ) INSTVER(9.3.0.0)
++ Cannot use MQ 9.3.0 administrative commands to run an MQ 9.1 queue manager
```

Important: You cannot use administrative commands on a different IBM MQ version from the one you are on.

If you attempt to do so you receive message AMQ5691E: Queue manager <qmname> is associated with a different installation (<installation name>).

Migrating a queue manager to a later version of IBM MQ

After installing IBM MQ 9.4, you now want to migrate and upgrade an IBM MQ 9.1 queue manager to be used with IBM MQ 9.4.

About this task

You need to perform two main steps:

1. Use the **setmqm** command to associate the queue manager with the desired installation, that is, the IBM MQ version.
2. Use the **strmqm** command under the desired version, which updates the data of the queue manager for the new version.

For this scenario, the queue manager QMMIG, which was created with IBM MQ 9.1, is going to be used as an illustration of the migration procedure.



Attention: Once a queue manager has been migrated to a later version of IBM MQ, it is no longer possible to use it with the previous version of IBM MQ. The migration process changes a number of files and object definitions, and it is not possible to go back.

Procedure

1. Take a backup of the queue manager using the **dmpmqcfg** command.

See [Backing up and restoring IBM MQ queue manager data](#) and [Backing up queue manager configuration](#) for more information.

- a) To specify all the attributes, including default ones, (except **setmqaut**, which is not included in the output, specify the following command:

```
dmpmqcfg -m QMgr -a > QMgr.dmpmqcfg.out.all.mqsc
```

- b) To capture the attributes in **setmqaut** format, issue the following command:

```
dmpmqcfg -m QMgr -o setmqaut > QMgr.dmpmqcfg.setmqaut.bat
```

Note: The output file containing the **setmqaut** commands, includes the name of the queue manager in each command. Therefore, if you want to restore the commands into a different queue manager, you need to edit the file and specify the desired queue manager name.

2. To restore the:

- a) Commands for **runmqsc**, issue:

```
runmqsc Qmgr < QMgr.dmpmqcfg.out.mqsc
```

or

```
runmqsc Qmgr < QMgr.dmpmqcfg.out.all.mqsc
```

b) **setmqaut** commands issue:

```
QMGr.dmpmqcfg.setmqaut.bat
```

3. The queue manager to be migrated is at IBM MQ 9.1, so you need to run the script that sets the running environment to IBM MQ 9.4:

```
C:\> set-mq-80
```

- a) Issue the command `C:\> dspmqver` to check the version that the queue manager is running on IBM MQ 9.1.
- b) Issue the command `C:\> where dspmq` to check that the queue manager is running:

```
C:\Program Files\IBM\WebSphere MQ\bin64\dspmq.exe
C:\Program Files\IBM\WebSphere MQ\bin\dspmq.exe
```

- c) Issue the command `C:\> dspmq -m QMMIG -o installation -s`

```
QMNAME(QMMIG)                STATUS(Running) INSTNAME(Installation1)
INSTPATH(C:\Program Files\IBM\WebSphere MQ) INSTVER(9.1.0.9)
```

- d) Issue the command `C:\> runmqsc QMMIG`

```
display qmgr cmdlevel version
1 : display qmgr cmdlevel
AMQ8408: Display Queue Manager details.
  QMNAME(QMMIG)                CMDLEVEL(910)
  VERSION(09001009)
end
```

and note that the CMDLEVEL is at IBM MQ 9.1.

- e) Stop the queue manager by issuing the command `C:\> endmqm -i QMMIG`

You receive the messages:

```
WebSphere MQ queue manager 'QMMIG' ending.
WebSphere MQ queue manager 'QMMIG' ended.
```

4. Change the environment to run the IBM MQ 9.4 commands, either by issuing the command `C:\> set-mq-930` if you created the batch file, or by using the **setmqenv** command, and then check the version by issuing the **dspmqver** command.
5. Display the status of the queue managers by issuing the command `C:\> dspmq -o installation -s`.

You receive the following output:

```
QMNAME(QM91)                STATUS(Running) INSTNAME(Installation1)
INSTPATH(C:\Program Files\IBM\WebSphere MQ) INSTVER(9.1.0.9)

QMNAME(QMMIG)                STATUS(Ended immediately) INSTNAME(Installation1)
INSTPATH(C:\Program Files\IBM\WebSphere MQ) INSTVER(9.1.0.9)

QMNAME(QM910)                STATUS(Running) INSTNAME(Installation2)
INSTPATH(C:\Program Files\IBM\MQ) INSTVER(9.3.0.0)
```



Attention: Queue manager QMMIG is still associated with Installation1 (IBM MQ 9.1) while Installation2 is associated with IBM MQ 9.4.

Therefore, you need to disassociate the queue manager QMMIG from Installation1 and associate it with Installation2

6. Issue the following command to associate queue manager QMMIG with Installation2

```
C:\> setmqm -m QMMIG -n Installation2
```

You receive the following message:

```
The setmqm command completed successfully.
```

which informs you that QMMIG is now associated with IBM MQ 9.4.

7. Start queue manager QMMIG by issuing the command `C:\> strmqm QMMIG`

As this is the first time that the IBM MQ 9.4 **strmqm** command is being issued on a queue manager that had prior use with an older version, a migration takes place.

You see output similar to the following:

```
IBM MQ queue manager 'QMMIG' starting.  
The queue manager is associated with installation 'Installation2'.  
5 log records accessed on queue manager 'QMMIG' during the log replay phase.  
Log replay for queue manager 'QMMIG' complete.  
Transaction manager state recovered for queue manager 'QMMIG'.  
Migrating objects for queue manager 'QMMIG'.  
Default objects statistics : 5 created. 0 replaced. 0 failed.  
IBM MQ queue manager 'QMMIG' started using V9.3.0.0.
```

8. Issue the following command, `C:\> runmqsc QMMIG` to display the attributes of the queue manager, and note the VERSION and CMDLEVEL fields:

```
display qmgr cmdlevel version  
1 : display qmgr cmdlevel version  
AMQ8408I: Display Queue Manager details.  
  QMNAME(QMMIG)                      CMDLEVEL(930)  
  VERSION(09300000)  
end
```

Results

You have successfully migrated a queue manager to a later version of the product.

Installing a fix pack on IBM MQ 9.4

How you install a fix pack on top of the installed IBM MQ 9.4 on a system that has multi-version installations of IBM MQ.

Before you begin

Ensure you have migrated the queue manager QMMIG to IBM MQ 9.4. See [“Migrating a queue manager to a later version of IBM MQ”](#) on page 147 for more information.

About this task

For this scenario, there is another installed version (IBM MQ 9.1) and the queue managers running under that other version will not be stopped in order to demonstrate that you can continue using those other versions, while performing maintenance activities for IBM MQ 9.4.0.n.

Note that no advanced options are selected when installing the update.

Procedure

1. Login as an administrator.
 - a) Issue the command `C:\> set-mq-930` if you created the batch file, or use the **setmqenv** command, to make sure you are on IBM MQ 9.4.0.n, where n is 0 in this scenario.
 - b) Display the status of the queue managers by issuing the command `C:\> dspmq -o installation -s`.

You receive the following output:

```
QMNAME(QM80)                      STATUS(Running)INSTNAME(Installation1)  
INSTPATH(C:\Program Files\IBM\MQ) INSTVER(9.1.0.0)  
  
QMNAME(QMMIG)                     STATUS(Ended unexpectedly)
```

```
INSTNAME(Installation2)
INSTPATH(C:\Program Files\IBM\MQ) INSTVER(9.3.0.0)

QMNAME(QM910) STATUS(Ended immediately) INSTNAME(Installation2)
INSTPATH(C:\Program Files\IBM\MQ) INSTVER(9.3.0.0)
```

2. Stop the IBM MQ 9.4.0 processes.

See [Applying and removing maintenance on Windows](#) for further information on stopping the processes.

In summary:

- Issue the **endmqm immediate** command to the queue managers QMMIG and QM930.
- Stop the IBM MQ service for the installation by right-clicking the IBM MQ icon in the taskbar, then click **Stop IBM MQ**.

3. Locate the fix pack. This scenario is using IBM MQ 9.4.0.5.

Go to [Recommended Fixes for IBM MQ](#) for a list of the latest:

- Continuous Delivery release, and the Fix List for Continuous Delivery releases
- Cumulative Security Update for the Long Term Support release and the Fix List for Long Term Support releases

Note: You should check you are using the latest fix pack by visiting and download the appropriate one.

a) Click on the appropriate tab.

For this scenario it is *V9.3.0.5 LTS*

b) Download the Windows software, from Fix Central or Passport Advantage. if you need to download the entire product,

In this scenario, you put the file set into the following directory C:\downloads\mq9305; the file name is 9.3.0-IBM-MQ-Win64-FP0005.zip.

4. Extract the files from the .zip file and run the following command, **IBM-MQ-9.3.0-FP0005.exe**.

You see the *Install Anywhere* Dialog, containing the following information *InstallAnywhere is preparing to install ...*

You need to wait until the preparation is over which can take several minutes.

5. Click **OK** to continue when the window, *IBM MQ (fix pack 9.3.0.5 files)* appears.

a) When the *Introduction* section appears, click **Next**.

b) When the *Installation Type* section appears, select the option that is more suitable for your enterprise, probably **Load files and apply fix pack**, and click **Next**.

c) When the *Information* section appears, click **Next**.

d) When the *Destination Folder* section appears, select the default location C:\Program Files\IBM\source\MQ 9.3.0.5, and click **Next**.

e) When the *Advanced Options* section appears, click **Next**.

f) When the *Pre-installation summary* section appears, check the information displayed, and click **Install**.

g) Wait while the code is loaded.

The *Loading* section has a progress indicator, and when the process is complete, *Loading Complete* is ticked, and the *Apply Fix Pack* section appears.

h) Click **Done**.

As there is more than one installation on your system, a dialog appears where you can select which installation you want to upgrade. In this case it is *Installation2 (9.3.0.0)*.

i) Click **OK**.

j) Accept the default for the *Backup Folder* and click **Apply**.

More dialogs appear with progress indicators, and the final dialog states *Fix Pack 9.3.0.5 has been applied. Click Finish to end*.

k) Click **Finish**.

Results

You have successfully upgraded a version of IBM MQ for Windows alongside an existing version of the product.

Managed File Transfer scenario

An introduction to common Managed File Transfer topologies, and a scenario that demonstrates use of the Managed File Transfer capability by showing how to set up the system and transfer a test message.

- [Common topologies](#)
- [Configuring the base server](#)

MFT common topologies

This section describes common Managed File Transfer topologies. The double-sided arrows in each diagram represent connections to the queue manager.

See [“Connectivity considerations” on page 154](#) for more information on queue manager connection options.

Base topology with one queue manager

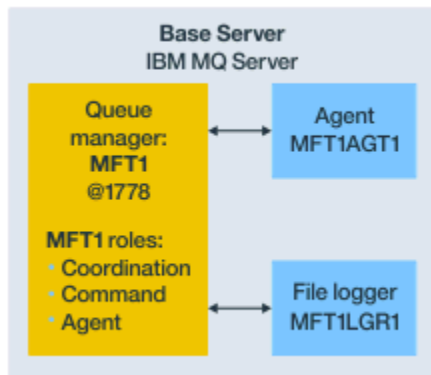


Figure 25. Base topology with one queue manager

A base topology represents a complete configuration which includes the coordination queue manager. The configuration name is the same as the name of the coordination queue manager. If the coordination queue manager name is MFT1, the configuration name is MFT1.

The base topology is the first Managed File Transfer configuration that you complete. After the base configuration is completed, partner agents from remote servers are added to the base configuration to exchange files.

The base topology does not exchange files outside the base topology server. However, the base topology enables you to move files to different locations in the same server and could be used for development purposes.

Base topology with one partner agent

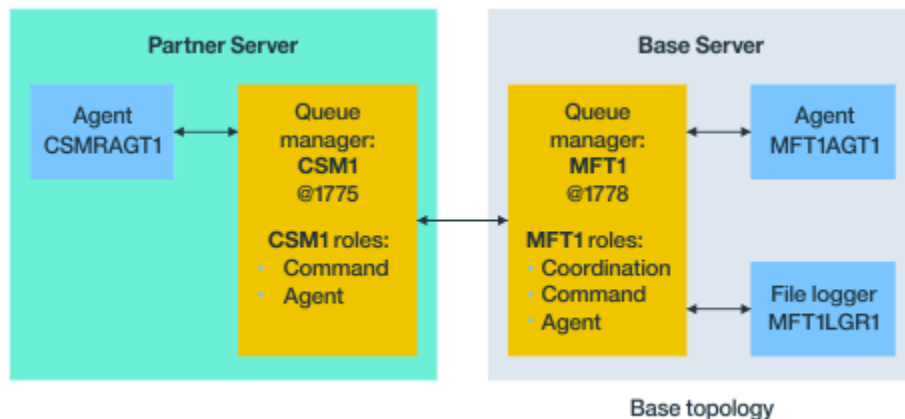


Figure 26. Base topology with one partner agent

This topology can exchange files between the two agents. Extra partner agents can be added in a similar way to the first added agent.

You can use a single queue manager for all three Managed File Transfer queue manager roles, or you can use dedicated queue managers for specific roles.

For example, you could have one queue manager dedicated to the coordination queue manager role, and the command and agent roles might share a second queue manager.

The connection between a remote agent queue manager in a separate server from the base configuration, and the base configuration coordination queue manager must be configured as an IBM MQ client, or MQI channel.

The connection to the coordination queue manager is established by the **fteSetupCoordination** command. If the coordination queue manager connection is not configured as an IBM MQ client channel, at the partner server, commands such as **fteListAgents** fail when issued from the partner agent server.

Base topology with separate coordination queue manager and one partner agent

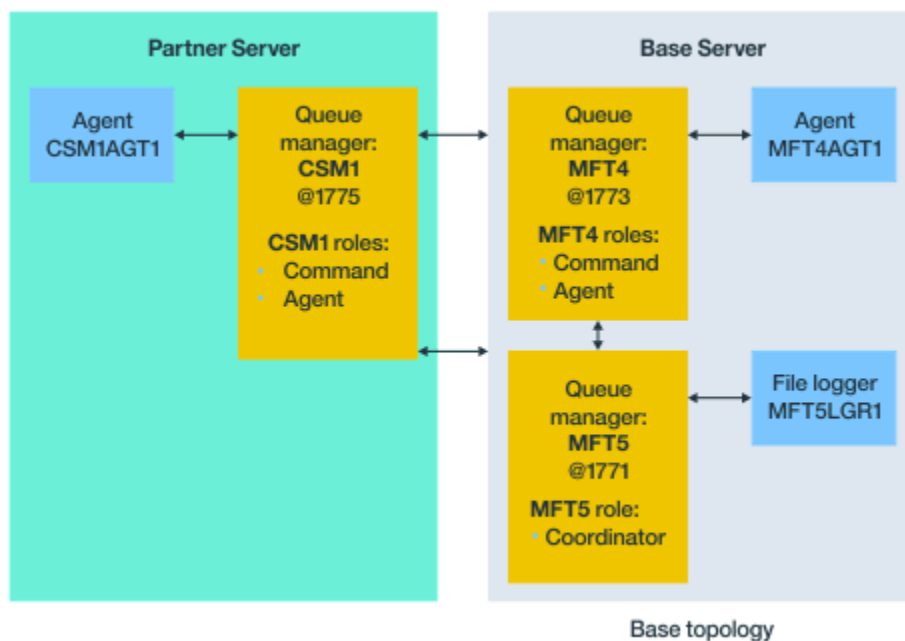


Figure 27. Base topology with separate coordination queue manager and one partner agent

In the base topology in Figure 3, on the base server, queue manager MFT4 is shared for the command and agent roles, and queue manager MFT5 is dedicated to the coordination queue manager role.

Connectivity must exist across all queue managers in the topology, including queue managers in the base topology, MFT4 and MFT5.

On the partner server queue manager, queue manager CSM1 has the roles of agent and commands queue manager.

This topology can exchange files between the two agents. Each partner agent must connect to a queue manager, as shown in the diagram. Extra partner agents can be added in a similar way, to the way that the first partner agent was added.

Base topology with Managed File Transfer Agent partner

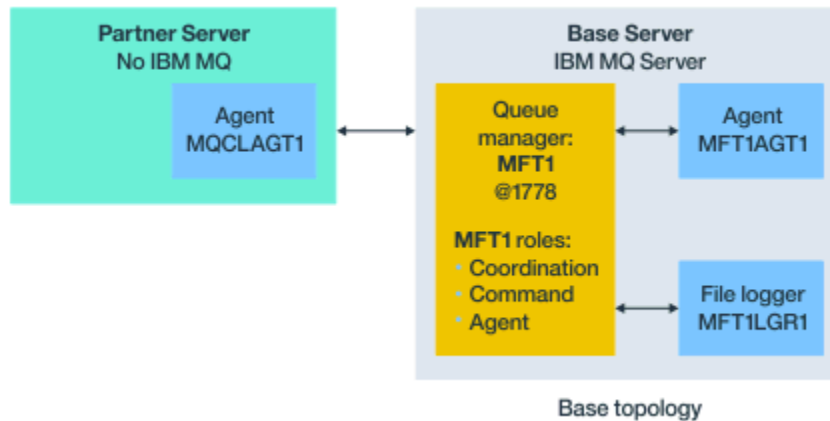


Figure 28. Base topology with Managed File Transfer Agent partner

This topology can exchange files between the two agents.

The server in the partner agent, depicted as MQCLAGT1 in the diagram, does not have IBM MQ server installed.

The partner agent is configured by using the same commands as the IBM MQ installed server, with some exceptions:

- The configuration for this partner agent must use IBM MQ client connections to base queue manager or queue managers.
- There is no need to run the coordination queue manager role IBM MQ definitions created by the configuration commands in the partner agent server. The coordination queue manager definitions already exist in the base server.

However, you must:

- Copy the agent object definitions generated when the agent is created in the partner server
- Transfer the definition file to the base configuration server, and
- Create the definitions in the queue manager identified as the agent queue manager in the base server.

In this case, MFT1 is serving all three roles, and you create the objects for agent MQCLAGT1 in the MFT1 queue manager.

As an alternative to copying the object definitions to the base server, you can run the **fteDefine** command for agent MQCLAGT1 on the base server where the agent queue manager is located. Use the definitions generated by the **fteDefine** command to create the required agent definitions on the agent queue manager.

For example, in the diagram shown, you would copy file `MQCLAGT1_create.mqsc` from the agent directory in the partner server, to the base configuration server, and create the required agent definitions in the MFT1 queue manager.

The configuration you complete on the partner agent server creates the Managed File Transfer configuration directory and required property files.

On the partner server you can install the [Managed File Transfer Redistributable Client](#) from Fix Central.

Note: The MQMFT redistributable client differs from the IBM MQ redistributable client, in that it is already packaged and does not require use of the **genmqpkg** utility. For more information, see [Redistributable Clients](#).

Connectivity considerations

In the preceding diagrams, each line across the agents and queue managers represents a connection to a queue manager.

This connection might be:

- A local connection
- A bindings or message channel connection, or
- An IBM MQ client or MQI connection.

The type of connection you select in your configuration depends on the parameters you specify

- When you specify the queue manager name parameter without other connection parameters, you specify a bindings connection.

If the queue manager used is local to the Managed File Transfer configuration, it also represents a local connection, when used in the base configuration server.

- If you specify the queue manager name parameter, along with the corresponding host, port, and channel name parameters, you specify an IBM MQ client connection.

When agents are located on the same host as the agent queue manager, a bindings type specification, which results in a local connection, is more efficient.

Configuring the base server

How you set up the base server with a separate configuration queue manager.

Before you begin

The following example assumes that you have:

- Reviewed the section [“Connectivity considerations” on page 154](#) and understand how to influence the type of connection to queue managers in the configuration.
- A working IBM MQ infrastructure. See [Configuring IBM MQ queue managers](#) for information on setting up queue managers.
- IBM MQ security tasks are completed.

All system resources, such as access to files, are configured with adequate security.

For Managed File Transfer security configuration, refer to [Securing Managed File Transfer and Restricting user authorities on MFT agent actions](#).

- All IBM MQ connections are tested after IBM MQ is configured by either using a sample program to send and receive messages, or using sample **amqscnxc** to test IBM MQ client type connections.

The **amqscnxc** sample connects to a queue manager by defining the channel connection in the sample code, which is similar to the way that Managed File Transfer connects, when it uses an MQI or IBM MQ client type connection.

- The instructions assume that the server you use for the base configuration has one IBM MQ version installed. If you have multiple IBM MQ installations in the base server, you must be careful to use the correct file path for the version of IBM MQ you want to use.
- The queue managers used in these instructions do not require connection authentication.

While it might be simpler to complete your first configuration without connection authentication required, if your enterprise requires immediate use of connection authentication, see [MFT and IBM MQ connection authentication](#) for instructions on how to configure an `MQMFTCredentials.xml` credentials file

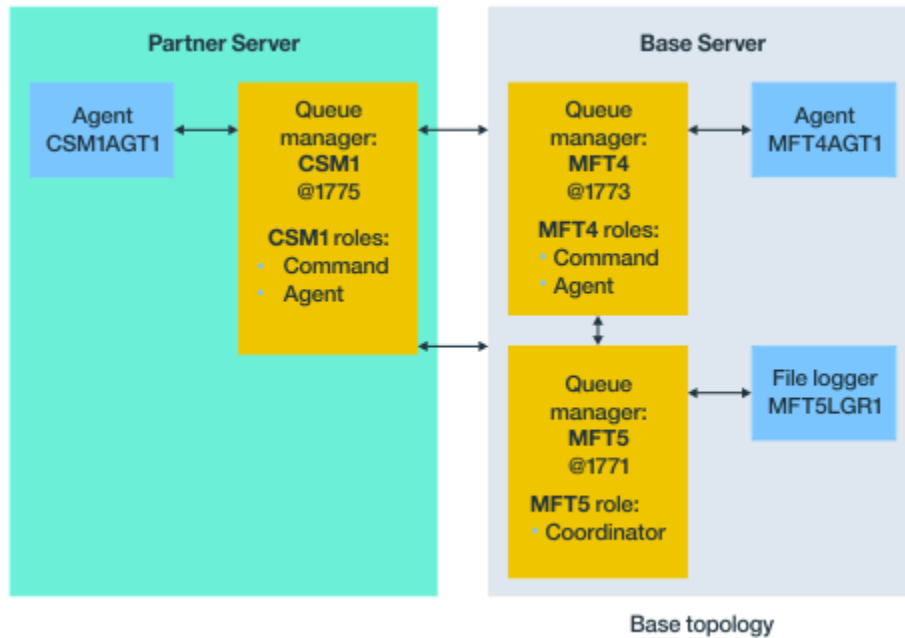


Figure 29. Base topology with separate coordination queue manager and one partner agent

About this task

The queue manager roles for the example configuration are:

- Base server
 - Queue manager MFT5 is the coordination queue manager
 - Queue manager MFT4 is used as the agent queue manager for agent MFT4AGT1, and also serves as the command queue manager for the MFT5 configuration on the base server.
- Partner server
 - Queue manager CSM1 doubles as the agent queue manager for agent CSM1AGT1, and as the command queue manager for the MFT5 configuration on the partner server.
 - Queue manager MFT5, on the base server, is the coordination queue manager.

Procedure

1. [Configure the coordination queue manager](#)
2. [Configure the command queue manager](#)
3. [Set up the agent](#)
4. [Set up the logger](#)
5. [Configure a partner server](#)

What to do next

Set up the [MQExplorer with MQMFT](#) so that you can test your example setup.

Configuring the coordination queue manager

How you configure the Coordination queue manager to coordinate file transfers.

Before you begin

Ensure that you have full connectivity between the queue managers you set up for this scenario.

About this task

This task sets up the coordination queue manager MFT5, and the instructions in this section assume that you are working with one IBM MQ installation.

If you have multiple installations, you must set the IBM MQ path to the version of IBM MQ required, by using the [setmqenv](#) command, before you start any of the configuration tasks.

Procedure

1. Log in as the Managed File Transfer administrator.
2. Issue the following command to identify the coordination queue manager and set up the configuration directory structure:

```
fteSetupCoordination -coordinationQMgr MFT5
```

Coordination queue manager directory

C:\data\mqft\config\MFT5

coordination.properties file

C:\data\mqft\config\MFT5\coordination.properties

The command also produces an MQSC command file that you must run against your coordination queue manager C:\data\mqft\config\MFT5\MFT5.mqsc:

3. Change to the C:\data\mqft\config\MFT5 directory.
4. Configure the queue manager, to act as the coordination queue manager, by running the following command.

You need to provide the MQSC command file, produced by the command you issued in Step “2” on [page 156](#):

```
runmqsc MFT5 < MFT5.mqsc > mft5.txt
```

5. Open the mft5.txt results file with your preferred editor. and ensure that the definitions have been created successfully.

What to do next

Set up the [command queue manager](#).

Configuring the commands queue manager

How you configure the commands queue manager.

Before you begin

Ensure that you have configured the coordination queue manager. See [“Configuring the coordination queue manager” on page 156](#) for more information.

About this task

This task identifies the commands queue manager.

Procedure

Issue the following command:

```
fteSetupCommands -connectionQMgr MFT4
```

You obtain the following message BFGCL0245I: The file C:\data\mqft\config\MFT4\command.properties has been created successfully.

The command queue manager does not require extran IBM MQ definitions. After you run **fteSetupCommands**, the command.properties file is created in the MFT5 configuration directory.

What to do next

Set up the [agent](#).

Setting up the agent

How you prepare a file transfer agent MFT4AGT1, including MQSC scripts that you must run.

Before you begin

You should have set up the command queue manager. See [“Configuring the commands queue manager”](#) on page 156 for more information.

About this task

This task prepares the Windows file transfer agent, MFT4AGT1.

Procedure

1. Issue the following command:

```
fteCreateAgent -agentName MFT4AGT1 -agentQMgr MFT4
```

After you create the agent with the **fteCreateAgent** command, the agents directory, and a subdirectory for the agent, MFT4AGT1, are added to the MFT5 directory.

In the *data\MFT5\agents\MFT4AGT1* directory you find the:

- agent.properties file
 - MFT4AGT1_create.mqsc file, which contains IBM MQ definitions required by the agent.
2. Change to the *data\MFT5\agents\MFT4AGT1* directory, and create the required agent queue manager definitions by issuing the following command:

```
runmqsc MFT4 < MFT4AGT1_create.mqsc > mft4.txt
```

3. Open the mft4.txt results file with your preferred editor and ensure that the definitions have been created successfully.
4. Start the agent by typing the following command: **fteStartAgent** MFT4AGT1.
5. Display the agent by typing the following command: **fteListAgents**.

You should see output similar to the following:

```
5655-MFT, 5724-H72 Copyright IBM Corp. 2008, 2024. ALL RIGHTS RESERVED  
BFGPR0127W: No credentials file has been specified to connect to IBM MQ.
```

```
Therefore, the assumption is that IBM MQ authentication has been disabled.  
Agent Name:      Queue Manager Name:      Status:  
MFT4AGT1        MFT4                      READY
```

Note: if you have not enabled connection authentication in your Managed File Transfer environment, you can ignore the BFGPR0127W message.

If you issue the **ftelistAgents** command and receive the following message, BFGCL0014W: No agents exist that match the current selection criteria., see [What to do if your MFT agent is not listed by the **ftelistAgents** command.](#)

What to do next

Set up the [logger](#).

Setting up the logger

A file or database logger is required to keep history and audit information about transfer activity for the configuration. In this example you create a file logger.

Before you begin

You must have set up the:

- Configuration queue manager
- Command queue manager
- Agent

Procedure

1. Issue the following command:

```
fteCreateLogger -loggerQMgr MFT5 -loggerType FILE  
-fileLoggerMode CIRCULAR -fileSize 5MB -fileCount 3 MFT5lgr1
```

After you run the **fteCreateLogger** command, the `data\mqft\config\MFT5\loggers` directory is created, with an MFT5LGR1 subdirectory.

The MFT5LGR1 subdirectory holds the `logger.properties` file. Also in the directory is a file called `MFT5LGR1_create.mqsc` with IBM MQ definitions required by the logger.

2. Change to the directory `data\mqft\config\MFT5\loggers\MFT5LGR1`.
3. Run the associated MQSC command file.

```
runmqsc MFT5 < MFT5_create.mqsc
```

to create the definitions required by the logger.

- a) Review the results of the object definitions to confirm that the required objects have been created successfully.
4. Start the logger by issuing the following command **fteStartLogger** MFT5LGR1.
 5. Review the contents of file `output0.log` at `data\mqft\logs\MFT5\loggers\MFT5LGR1\logs`.

After some information about the logger, the last statement should contain message: BFGDB0023I: The logger has completed startup activities and is now running.

Occasionally, log information might not be written to the `output0.log` the first time the logger starts. If the `output0.log` file is empty, restart the logger by typing **fteStopLogger** MFT5LGR1 and pressing the **Enter** key.

Restart the logger by typing **fteStartLogger** MFT5LGR1 and pressing the **Enter** key. File `output0.log` now shows data.

The same behavior extends to the agent version of the `output0.log` file the first time an agent is started.

Stop and start the agent by using **fteStopAgent** and **fteStartAgent** commands. You then see log data written to the agent `output0.log` file.

Results

You have configured the base server, which includes the coordination queue manager for this configuration.

What to do next

You now do similar work for the partner server, which contains a remote agent.

Configuring a partner server

How you configure a partner server, when the base server has a separate coordination queue manager

Before you begin

Ensure that you have fully completed all the tasks to set up a base server, that includes a configuration queue manager.

About this task

The same assumptions made about IBM MQ and the security configuration, as well as the IBM MQ path also apply to the partner server.

Start by setting up the MFT5 configuration directory, and identifying the coordination queue manager by using the **fteSetupCoordination** command.

Procedure

1. Create the partner server configuration directory by issuing the following command:

```
fteSetupCoordination -coordinationQMGr MFT5  
-coordinationQMGrHost 177.16.20.15 -coordinationQMGrPort 1771  
-coordinationQMGrChannel MQMFT.MFT5.SVRCONN
```

Notes:

- a. When the coordination queue manager is on a different server from the partner server, the connection to the base server coordination queue manager must be defined as a client connection.

Failure to define the coordination queue manager connection as an IBM MQ client connection, on the partner server, causes any Managed File Transfer command, that connects to the coordination queue manager, to fail.

An example of a command that connects to the coordination queue manager is **fteListAgents**.
 - b. You do not need to create the IBM MQ definitions as the definitions required by the coordination queue manager were completed when you configured the base server.
2. Identify the commands queue manager by issuing the following command:

```
fteSetupCommands -connectionQMGr CSM1
```

The commands queue manager does not require any extran IBM MQ definitions.

3. Identify the partner agent queue manager, and create the partner agent queue manager, by issuing the following command:

```
fteCreateAgent -agentName CSM1AGT1 -agentQMgr CSM1
```

4. Change to the CSM1AGT1 directory.
5. Create the IBM MQ definitions required by the agent, by issuing the following command:

```
runmqsc CSM1 < CSM1AGT1_create.mqsc > csm1.txt
```

- a) Open file csm1.txt with your preferred editor to confirm that all agent required definitions have been created successfully.
6. Start the agent, by issuing the following command:

```
fteStartAgent CSM1AGT1
```

7. Display the agent by typing **fteListAgents**

You should see output similar to the following:

```
C:\>fteListAgents
5655-MFT, 5724-H72 Copyright IBM Corp. 2008, 2024. ALL RIGHTS RESERVED
BFGPR0127W: No credentials file has been specified to connect to IBM MQ. Therefo
re, the assumption is that IBM MQ authentication has been disabled.
Agent Name:      Queue Manager Name:    Status:
CSM1AGT1         CSM1                                READY
MFT4AGT1         MFT4                                READY
```

Note: if you have not enabled connection authentication in your Managed File Transfer environment, you can ignore the BFGPR0127W message.

If you issue the **ftelistAgents** command and receive the following message, BFGCL0014W: No agents exist that match the current selection criteria., see [What to do if your MFT agent is not listed by the **fteListAgents** command.](#)

If the status of one of the agents is UNREACHABLE, see [What to do if an agent is shown as being in an UNKNOWN state.](#)

Setting up the IBM MQ Explorer with MFT

This task helps you connect IBM MQ Explorer to the Managed File Transfer configuration.

Procedure

1. Start IBM MQ Explorer.
2. In the left Navigator panel, scroll down and expand the folder: Managed File Transfer.
You see the entry for the Coordination queue manager: MFT5
3. Right click on MFT5 and select **Connect**.
 - a) Select Agents in the drop down menu that appears and ensure that both agents, MFT4AGT1 and CSMAGT1, are in the Ready state.

What to do next

Test your example setup with [IBM MQ Explorer](#).

Using the IBM MQ Explorer to test a file transfer

This task gives an example of how you use IBM MQ Explorer with Managed File Transfer, to test a file transfer, after you have set up the IBM MQ Explorer as described in the previous topic.

Before you begin

Ensure that you have a working system, that the agents are READY and IBM MQ Explorer is working. See [“Setting up the IBM MQ Explorer with MFT” on page 160](#) for more information.

About this task

Determine the file to use to test the transfer, and a directory to copy it to. For this example, it is assumed that file `test-file.txt` out of directory `C:\temp\mft` is used.

```
C:\temp\mft> dir *  
Date stamp 61 test-file.txt  
1 File(s) 61 bytes
```

Procedure

1. Start the IBM MQ Explorer in Windows
2. In the left Navigator panel, expand the folder: Managed File Transfer.
You see the entry for the Coordination queue manager: MFT5
3. Right click on MFT5 and select **Connect**.
4. Once connected, right click on MFT5 and select **New Transfer**
 - a) Use the pull down menu to select MFT4AGT1 for the Source agent and CSMAGT1 for the Destination agent.
 - b) Click **Next**.
 - c) Click **Add** on the next window.
A wide dialog appears. The left side is for Source and the right side for Destination.
5. On the Source panel:
 - a) Select **Text transfer** as the file is text.
 - b) Select **Browse** to locate the file.
In this case, the file is `C:\temp\mft\test-file.txt`.



Attention: Do not click **OK** as you need to complete the Destination panel.

6. On the Destination panel:
 - a) Enter the name that you are giving the file at the destination, for example, `test-file.txt`.
The use of relative paths is supported. The top portion of the full path is the home directory of the user ID who starts the destination agent.
 - b) Select **Overwrite files if present** if you require this option.
 - c) Click **OK**.
The file you selected appears in the **New Transfers** panel.
7. If the MFT5 configuration menu is closed, and shows +MFT5, expand the menu by clicking the + sign.
8. Stay at the MFT configuration selected.
Next, you check the status of the transfer by carrying out the following procedure.
9. Click **Transfer Log** under the coordination queue manager MFT5.
10. Look at the status in Managed File Transfer - Current Transfer progress panel, immediately below the **Transfer Log** top panel and wait for the transfer to complete.

If the transfer shows successful and with a green background, you have successfully completed the test of your configuration.

If the transfer failed with a red background, an error occurred.

In most cases, you can use the scroll bar below the top **Transfer Log** panel and view a summary of the reasons for failure.

- a) If you cannot determine why the transfer failed, double-click the entry for the transfer in the top **Transfer Log** panel.
- b) Select XML in the left pane of the pop-up panel that appears.
- c) Scroll through the information to determine the cause of the error.
- d) Make the corrections needed and test the transfer again.

Getting started with IBM MQ Internet Pass-Thru

These scenarios show you how to set up some simple IBM MQ Internet Pass-Thru (MQIPT) configurations. You can also use these tasks to confirm that the product has been installed successfully.

Before you begin

Before you start to use these scenarios, make sure that the following prerequisites have been completed:

- You are familiar with defining queue managers, queues, and channels on IBM MQ.
- You have already installed an IBM MQ client and server.
- MQIPT is installed in a directory called C:\mqipt on Windows systems. (The examples are written for Windows systems but will run on any of the supported platforms.) For more information on installing MQIPT, see [Installing MQIPT](#).
- The client, server, and each instance of MQIPT are installed on separate computers.
- You are familiar with putting messages on a queue by using the **amqsputc** command.
- You are familiar with getting messages from a queue using the **amqsgetc** command.
- You are familiar with setting client authorities in IBM MQ.

About this task

After completing the prerequisites, complete the following initial steps to set up the system ready for working through the scenarios.

Procedure

1. On the IBM MQ server, complete the following tasks:

- Define a queue manager called MQIPT.QM1.
- Define a server connection channel called MQIPT.CONN.CHANNEL.
- Define a local queue called MQIPT.LOCAL.QUEUE.
- Start a TCP/IP listener for MQIPT.QM1 on port 1414. If port 1414 is already in use by another application choose a free port address and substitute it in the following examples.
- Ensure that connection authentication and channel authentication is configured to allow client connections from the client machine with your user ID. If connection authentication is set to require applications to supply authentication credentials for client connections, you will need to set one of the following environment variables before running the **amqsputc** and **amqsgetc** commands:

MQSAMP_USER_ID

Set to the user ID to be used for connection authentication, if you want use a user ID and a password to authenticate with the queue manager.

Set to a non-blank value if you want to use an authentication token to authenticate with the queue manager.

2. Test the route from the IBM MQ client to the queue manager by putting a message on the local queue of the queue manager, by using the **amqsputc** command, and then retrieving it, by using the **amqsgetc** command.

To prepare for the scenarios in this section, create and edit the `mqipt.conf` file as follows:

- a. Copy the `mqiptSample.conf` file, which you can find in the `samples` subdirectory of the MQIPT installation directory, to `mqipt.conf` in your chosen MQIPT home directory. The following scenarios use `C:\mqiptHome` as the MQIPT home directory.
- b. Create two directories alongside `mqipt.conf` named `errors` and `logs`. Set the file permissions on these directories so that they are writeable by the user ID that will run MQIPT.
- c. Delete all routes from the `mqipt.conf` file.
- d. In the remaining `[global]` section, check that **ClientAccess** exists and is set to `true`.

What to do next

After setting up your system, you are ready to start the following scenarios:

- [“Verifying that MQIPT is working correctly” on page 163](#)
- [“Request a CA-signed certificate for MQIPT” on page 166](#)
- [“Creating test certificates” on page 165](#)
- [“Authenticating a TLS server” on page 168](#)
- [“Authenticating a TLS client” on page 170](#)
- [“Configuring HTTP tunneling” on page 176](#)
- [“Configuring access control” on page 178](#)
- [“Configuring a SOCKS proxy” on page 180](#)
- [“Configuring a SOCKS client” on page 182](#)
- [“Configuring MQIPT clustering support” on page 184](#)
- [“Allocating port numbers” on page 187](#)
- [“Retrieving CRLs by using an LDAP server” on page 188](#)
- [“Running MQIPT in TLS proxy mode” on page 191](#)
- [“Running MQIPT in TLS proxy mode with a security manager” on page 192](#)
- [“Using a security exit” on page 195](#)
- [“Routing client connection requests to IBM MQ queue manager servers by using security exits” on page 197](#)
- [“Dynamically routing client connection requests” on page 200](#)
- [“Using a certificate exit to authenticate a TLS server” on page 203](#)

Verifying that MQIPT is working correctly

Use this simple configuration setup to ensure that MQIPT is installed correctly.

Before you begin

- Before you start to use this scenario, make sure that you have completed the prerequisite tasks listed in [“Getting started with IBM MQ Internet Pass-Thru” on page 162](#).

About this task

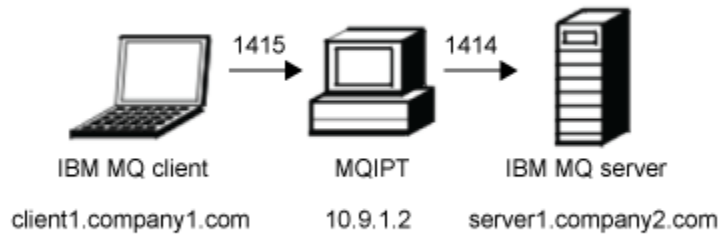


Figure 30. Installation verification test network diagram

This diagram shows the connection from the IBM MQ client (called client1.company1.com on port 1415) through MQIPT to the IBM MQ server (called server1.company2.com on port 1414).

Procedure

To verify that MQIPT is working correctly, complete the following steps:

1. Define an MQIPT route.

On the MQIPT computer, edit `mqipt.conf` and add a route definition:

```
[route]
ListenerPort=1415
Destination=server1.company2.com
DestinationPort=1414
```

2. Start MQIPT.

Open a command prompt and enter the following command:

```
C:\mqipt\bin\mqipt C:\mqiptHome -n ipt1
```

where `C:\mqiptHome` indicates the location of the MQIPT configuration file, `mqipt.conf`, and `ipt1` is the name to be given to the instance of MQIPT.

The following messages indicate that MQIPT has started successfully:

```
5724-H72 (C) Copyright IBM Corp. 2000, 2024. All Rights Reserved
MQCPI001 IBM MQ Internet Pass-Thru V9.4.0.0 starting
MQCPI004 Reading configuration information from mqipt.conf
MQCPI152 MQIPT name is ipt1
MQCPI021 Password checking has been enabled on the command port
MQCPI011 The path C:\mqiptHome\logs will be used to store the log files
MQCPI006 Route 1415 has started and will forward messages to:
MQCPI034 ....server1.company2.com(1414)
MQCPI035 ....using MQ protocol
MQCPI078 Route 1415 ready for connection requests
```

3. At a command prompt on the IBM MQ client system, enter the following commands:

- a) Set the **MQSERVER** environment variable:

```
SET MQSERVER=MQIPT.CONN.CHANNEL/tcp/10.9.1.2(1415)
```

- b) Put a message:

```
amqsputc MQIPT.LOCAL.QUEUE MQIPT.QM1
Hello world
```

Press the Enter key twice after typing the message string.

- c) Get the message:

```
amqsgetc MQIPT.LOCAL.QUEUE MQIPT.QM1
```

The message, "Hello world" is returned.

4. To stop IBM MQ, enter the following command:

```
mqiptAdmin -stop -n ipt1
```

Creating test certificates


A self-signed certificate can be used by an MQIPT route to identify itself to a remote peer.

Self-signed certificates can be useful in test scenarios where you must ensure TLS connectivity without paying a certificate authority (CA) for a certificate. However, do not use self-signed certificates in production environments. If you need to request a certificate to use in a production environment, see [“Request a CA-signed certificate for MQIPT” on page 166](#).

Before you begin

Before you start this task, complete the steps that are listed in [“Getting started with IBM MQ Internet Pass-Thru” on page 162](#).

About this task

 Use the **mqiptKeytool** command to create the certificate. Specify the name of the keystore that contains the certificate in the **SSLServerKeyRing** or **SSLClientKeyRing** MQIPT route property, depending on whether the certificate is used by inbound or outbound connections.

Procedure

Enter the following command to create a self-signed personal certificate for testing purposes in a PKCS #12 keystore:

```
mqiptKeytool -genkeypair -keystore keystore_name -storetype pkcs12 -storepass password  
-alias label -dname DN_identity  
-keyalg key_algorithm -keysize key_size -sigalg signature_algorithm
```

where:

-keystore *keystore_name*

Specifies the name of the keystore. For example, `mqiptKeys.p12`. The keystore is created if it does not exist.

-storepass *password*

Specifies the keystore password.

-alias *label*

Specifies the certificate label.

-dname *DN_identity*

Specifies the X.500 Distinguished Name for the certificate enclosed in double quotation marks. For example, `"CN=Test Certificate,OU=Sales,O=Example,C=US"`.

-keyalg *key_algorithm*

Specifies the algorithm that is used to create the key pair. For example, RSA.

-keysize *key_size*

Specifies the key size. For example, 2048.

-sigalg *signature_algorithm*

Specifies the algorithm that is used to sign the certificate. For example, SHA256WithRSA.

If you use the example values, this command creates a digital certificate with a 2048-bit RSA public key and a digital signature that uses RSA with the SHA-256 hash algorithm.

Choose an appropriate public key encryption algorithm, key size, and digital signature algorithm for your organization's security needs. For more information, see [Digital certificate considerations for MQIPT](#).

What to do next

Issue the following command to encrypt the keystore password:

```
mqiptPW
```



Enter the keystore password to encrypt when prompted. Set the value of the appropriate property in the **mqipt.conf** configuration file to the encrypted password that is output by the **mqiptPW** command. Depending on whether the certificate is used by inbound or outbound connections, set the value of either the **SSLServerKeyRingPW** or **SSLClientKeyRingPW** property or the encrypted password. For more information about encrypting keystore passwords, see [Encrypting stored passwords](#).

Request a CA-signed certificate for MQIPT



Request a certificate that is signed by a trusted certificate authority (CA) to allow MQIPT to use TLS.

Before you begin

Before you start this task, complete the steps that are listed in [“Getting started with IBM MQ Internet Pass-Thru”](#) on page 162.

  This task assumes that you request a new certificate from a CA by using the **mqiptKeytool** command, and that your personal certificate is returned to you in a file. One certificate is sufficient to enable server authentication for TLS connections made to MQIPT routes. If you require client authentication for TLS connections that are made by MQIPT to the route destination, request a second certificate by completing steps in this task twice to create two keystores.

About this task

  Use the **mqiptKeytool** command to create the certificate request, and to receive the signed certificate into the keystore. Specify the name of the keystore that contains the certificate in the **SSLServerKeyRing** or **SSLClientKeyRing** MQIPT route property, depending on whether the certificate is used by inbound or outbound connections.

Procedure

1. Enter the following command to create a key pair and an associated certificate in a PKCS #12 keystore:

```
mqiptKeytool -genkeypair -keystore keystore_name -storetype pkcs12 -storepass password  
-alias label -dname DN_identity  
-keyalg key_algorithm -keysize key_size -sigalg signature_algorithm
```

where:

-keystore *keystore_name*

Specifies the name of the keystore. For example, `mqiptKeys.p12`. The keystore is created if it does not exist.

-storepass *password*

Specifies the keystore password.

-alias *label*

Specifies the certificate label.

-dname *DN_identity*

Specifies the X.500 Distinguished Name for the certificate enclosed in double quotation marks. For example, `"CN=Test Certificate,OU=Sales,O=Example,C=US"`.

-keyalg *key_algorithm*

Specifies the algorithm that is used to create the key pair. For example, RSA.

-keysize *key_size*

Specifies the key size. For example, 2048.

-sigalg *signature_algorithm*

Specifies the algorithm that is used to sign the certificate. For example, SHA256WithRSA.

If you use the example values, this command creates a digital certificate with a 2048-bit RSA public key and a digital signature that uses RSA with the SHA-256 hash algorithm.

Choose an appropriate public key encryption algorithm, key size, and digital signature algorithm for your organization's security needs. For more information, see [Digital certificate considerations for MQIPT](#).

2. Enter the following command to create a certificate signing request:

```
mqiptKeytool -certreq -keystore keystore_name -storetype pkcs12 -storepass password
             -alias label -file certreq_filename
```

where:

-keystore *keystore_name*

Specifies the name of the keystore. For example, mqiptKeys.p12. The keystore is created if it does not exist.

-storepass *password*

Specifies the keystore password.

-alias *label*

Specifies the certificate label.

-file *certreq_filename*

Specifies the file name for the certificate request.

Send the certificate request file that the command creates to your CA to be signed.

3. When you receive the signed certificate from the CA, enter the following command to add the certificate to the keystore:

```
mqiptKeytool -importcert -keystore keystore_name -storetype pkcs12 -storepass password
             -file cert_filename
```

where:

-keystore *keystore_name*

Specifies the name of the keystore. For example, mqiptKeys.p12. The keystore is created if it does not exist.

-storepass *password*

Specifies the keystore password.

-file *cert_filename*

Specifies the name of the file that contains the signed certificate.

4. Ensure that the CA certificate of the CA that signed the personal certificate is present in the CA keystore. You can choose whether to add the CA certificate to the same keystore as the personal certificate, or to a separate keystore that is used only for CA certificates.

To use a separate CA keystore, you can either use the sample CA keystore named `sslCAdefault.pfx` that is supplied with MQIPT, or create a new PKCS #12 keystore. Add the public CA certificate of the CA that signed your personal certificate to the CA keystore, unless it is already present in the sample keystore.

The public CA certificate might be returned with your personal certificate. If the public CA certificate is not returned with your certificate, you must request the CA certificate from the same CA that supplied your personal certificate and then add it to the keystore.

Enter the following command to add the CA certificate to the CA keystore:

```
mqiptKeytool -importcert -keystore keystore_name -storetype pkcs12 -storepass password
             -file cert_filename -alias label
```

where:

-keystore *keystore_name*

Specifies the name of the CA keystore. The keystore is created if it does not exist.

-storepass *password*

Specifies the CA keystore password.

-file *cert_filename*

Specifies the name of the file that contains the CA certificate.

-alias *label*

Specifies the label to be given to the CA certificate

What to do next

Issue the following command to encrypt the keystore password:

```
mqiptPW
```

Enter the keystore password to encrypt when prompted. Set the value of the appropriate property in the **mqipt.conf** configuration file to the encrypted password that is output by the **mqiptPW** command. Depending on whether the certificate is used by inbound or outbound connections, set the value of either the **SSLServerKeyRingPW** or **SSLClientKeyRingPW** property or the encrypted password. For more information about encrypting keystore passwords, see [Encrypting stored passwords](#).

To use these new keystores for server authentication, place the keystores in a directory named `ssl` under the MQIPT home directory and set the following route properties:

```
SSLClientCAKeyRing=C:\\mqiptHome\\ssl\\sslCAdefault.pfx
SSLClientCAKeyRingPW=encrypted_password
SSLServerKeyRing=C:\\mqiptHome\\ssl\\myServer.pfx
SSLServerKeyRingPW=encrypted_password
SSLServerCAKeyRing=C:\\mqiptHome\\ssl\\sslCAdefault.pfx
SSLServerCAKeyRingPW=encrypted_password
```

For more information about configuring MQIPT to use TLS, see [“Authenticating a TLS server”](#) on page 168.

Authenticating a TLS server

In this scenario, you can test a TLS connection by using the test certificate in the sample (`sslSample.pfx`) key ring file, provided with MQIPT in the `samples/ssl` subdirectory of the MQIPT installation directory.

Before you begin

Before you start to use this scenario, make sure that you have completed the prerequisite tasks listed in [“Getting started with IBM MQ Internet Pass-Thru”](#) on page 162, and have read the topic [SSL/TLS support in MQIPT](#).

About this task

The connection is made between an IBM MQ client and a IBM MQ server through two instances of MQIPT. The connection between MQIPT 1 and MQIPT 2 uses TLS, with MQIPT 1 acting as the TLS client, and MQIPT 2 acting as the TLS server.

During the TLS handshake, the server sends its test certificate to the client and the client uses its copy of the certificate with the trust-as-peer flag set to authenticate the server. The CipherSuite `SSL_RSA_WITH_AES_256_CBC_SHA256` is used. The `mqipt.conf` configuration file in this scenario is based on the configuration file created in the [“Verifying that MQIPT is working correctly”](#) on page

163 scenario. For details on how to create a test certificate to use in this example, see [“Creating test certificates”](#) on page 165.

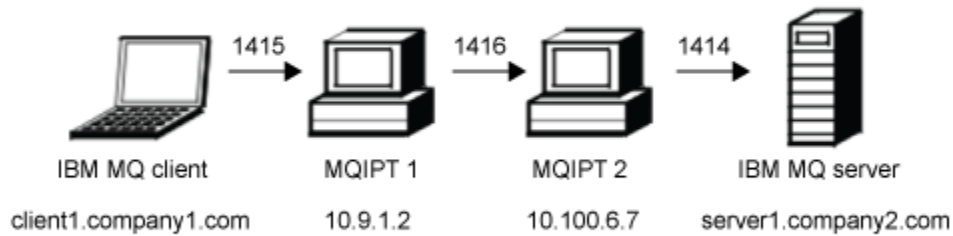


Figure 31. SSL/TLS server network diagram

This diagram shows the connection from the IBM MQ client (called client1.company1.com on port 1415) through two instances of MQIPT to the IBM MQ server (called server1.company2.com on port 1414).

Procedure

To authenticate a TLS server, complete the following steps:

1. On the MQIPT 1 system:

a) Edit mqipt.conf and add the following route definition:

```
[route]
ListenerPort=1415
Destination=10.100.6.7
DestinationPort=1416
SSLClient=true
SSLClientKeyRing=C:\mqipt\samples\ssl\sslSample.pfx
SSLClientKeyRingPW=<mqiptPW>1!PCaB1HWrFM0p43ngjwgArg==!6N/vsbqru7iqMhFN+wozxQ==
SSLClientCipherSuites=SSL_RSA_WITH_AES_256_CBC_SHA256
```

b) Open a command prompt and start MQIPT:

```
C:\mqipt\bin\mqipt C:\mqiptHome -n ipt1
```

where C:\mqiptHome indicates the location of the MQIPT configuration file, mqipt.conf, and ipt1 is the name to be given to the instance of MQIPT.

The following messages indicate that MQIPT has started successfully:

```
5724-H72 (C) Copyright IBM Corp. 2000, 2024. All Rights Reserved
MQCPI001 IBM MQ Internet Pass-Thru V9.4.0.0 starting
MQCPI004 Reading configuration information from mqipt.conf
MQCPI152 MQIPT name is ipt1
MQCPI021 Password checking has been enabled on the command port
MQCPI011 The path C:\mqiptHome\logs will be used to store the log files
MQCPI006 Route 1415 is starting and will forward messages to :
MQCPI034 ....10.100.6.7(1416)
MQCPI035 ....using MQ protocol
MQCPI036 ....SSL Client side enabled with properties :
MQCPI139 .....secure socket protocols <NULL>
MQCPI031 .....cipher suites SSL_RSA_WITH_AES_256_CBC_SHA256
MQCPI032 .....key ring file C:\mqipt\samples\ssl\sslSample.pfx
MQCPI047 .....CA key ring file <NULL>
MQCPI071 .....site certificate uses
UID=*,CN=*,T=*,OU=*,DC=*,O=*,STREET=*,L=*,ST=*,PC=*,C=*,DNQ=*
MQCPI038 .....peer certificate uses
UID=*,CN=*,T=*,OU=*,DC=*,O=*,STREET=*,L=*,ST=*,PC=*,C=*,DNQ=*
MQCPI078 Route 1415 ready for connection requests
```

2. On the MQIPT 2 system:

a) Edit mqipt.conf and add the following route definition:

```
[route]
ListenerPort=1416
Destination=Server1.company2.com
DestinationPort=1414
SSLServer=true
SSLServerKeyRing=C:\\mqipt\\samples\\ssl\\sslSample.pfx
SSLServerKeyRingPW=<mqiptPW>1!PCaB1HWrFM0p43ngjwgArg==!6N/vsbqru7iqMhFN+wozxQ==
SSLServerCipherSuites=SSL_RSA_WITH_AES_256_CBC_SHA256
```

- b) Open a command prompt and start MQIPT:

```
C:
cd \mqipt\bin
mqipt .. -n ipt2
```

where .. indicates that the MQIPT configuration file, `mqipt.conf`, is in the parent directory, and `ipt2` is the name to be given to the instance of MQIPT.

The following messages indicate that MQIPT has started successfully:

```
5724-H72 (C) Copyright IBM Corp. 2000, 2024. All Rights Reserved
MQCPI001 IBM MQ Internet Pass-Thru V9.4.0.0 starting
MQCPI004 Reading configuration information from mqipt.conf
MQCPI152 MQIPT name is ipt2
MQCPI021 Password checking has been enabled on the command port
MQCPI011 The path C:\mqipt\logs will be used to store the log files
MQCPI006 Route 1416 is starting and will forward messages to :
MQCPI034 ....Server1.company2.com(1414)
MQCPI035 ....using MQ protocol
MQCPI037 ....SSL Server side enabled with properties :
MQCPI139 .....secure socket protocols <NULL>
MQCPI031 .....cipher suites SSL_RSA_WITH_AES_256_CBC_SHA256
MQCPI032 .....key ring file C:\\mqipt\\samples\\ssl\\sslSample.pfx
MQCPI047 .....CA key ring file <NULL>
MQCPI071 .....site certificate uses
UID=*,CN=*,T=*,OU=*,DC=*,O=*,STREET=*,L=*,ST=*,PC=*,C=*,DNQ=*
MQCPI038 .....peer certificate uses
UID=*,CN=*,T=*,OU=*,DC=*,O=*,STREET=*,L=*,ST=*,PC=*,C=*,DNQ=*
MQCPI033 .....client authentication set to false
MQCPI078 Route 1416 ready for connection requests
```

3. At a command prompt on the IBM MQ client, enter the following commands:

- a) Set the **MQSERVER** environment variable:

```
SET MQSERVER=MQIPT.CONN.CHANNEL/tcp/10.9.1.2(1415)
```

- b) Put a message:

```
amqsputc MQIPT.LOCAL.QUEUE MQIPT.QM1
Hello world
```

Press Enter twice after typing the message string.

- c) Get the message:

```
amqsgetc MQIPT.LOCAL.QUEUE MQIPT.QM1
```

The message, "Hello world" is returned.

Authenticating a TLS client

In this scenario, you can test a TLS connection by using the sample test certificate to perform server and client authentication.

Before you begin

Before you start to use this scenario, make sure that you have completed the prerequisite tasks listed in [“Getting started with IBM MQ Internet Pass-Thru” on page 162](#), and have read the topic [SSL/TLS support in MQIPT](#).

About this task

The connection is made between an IBM MQ client and an IBM MQ server through two instances of MQIPT. The connection between MQIPT 1 and MQIPT 2 uses TLS, with MQIPT 1 acting as the TLS client, and MQIPT 2 acting as the TLS server.

During the TLS handshake, the server sends its test certificate to the client. The client uses its copy of the certificate, with the trust-as-peer flag, to authenticate the server. The client then sends its test certificate to the server. The server uses its copy of the certificate, with the trust-as-peer flag, to authenticate the client. The CipherSuite SSL_RSA_WITH_AES_256_CBC_SHA256 is used. The mqipt.conf configuration file in this scenario is based on the configuration file created in the [“Verifying that MQIPT is working correctly”](#) on page 163 scenario.

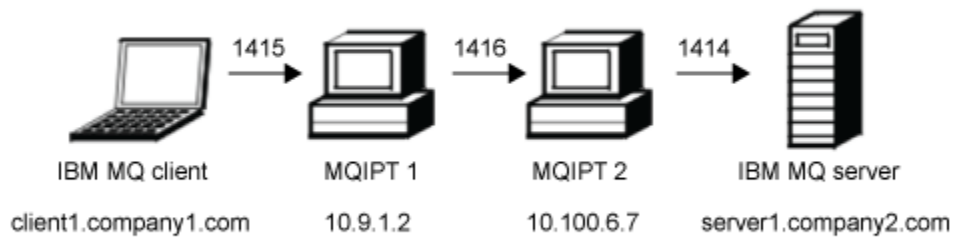


Figure 32. SSL/TLS client network diagram

This diagram shows the connection from the IBM MQ client (called client1.company1.com on port 1415) through two instances of MQIPT to the IBM MQ server (called server1.company2.com on port 1414).

Procedure

To authenticating a TLS client, complete the following steps:

1. On the MQIPT 1 system:

a) Edit mqipt.conf and add the following route definition:

```
[route]
ListenerPort=1415
Destination=10.100.6.7
DestinationPort=1416
SSLClient=true
SSLClientKeyRing=C:\mqipt\samples\ssl\sslSample.pfx
SSLClientKeyRingPW=<mqiptPW>1!PCaB1HWrfMOp43ngjwgArg==!6N/vsbqr7iqMhFN+wozQ==
SSLClientCipherSuites=SSL_RSA_WITH_AES_256_CBC_SHA256
```

b) Open a command prompt and start MQIPT:

```
C:\mqipt\bin\mqipt C:\mqiptHome -n ipt1
```

where C:\mqiptHome indicates the location of the MQIPT configuration file, mqipt.conf, and ipt1 is the name to be given to the instance of MQIPT.

The following messages indicate that MQIPT has started successfully:

```
5724-H72 (C) Copyright IBM Corp. 2000, 2024. All Rights Reserved
MQCPI001 IBM MQ Internet Pass-Thru V9.4.0.0 starting
MQCPI004 Reading configuration information from mqipt.conf
MQCPI152 MQIPT name is ipt1
MQCPI021 Password checking has been enabled on the command port
MQCPI011 The path C:\mqiptHome\logs will be used to store the log files
MQCPI006 Route 1415 is starting and will forward messages to :
MQCPI034 ....10.100.6.7(1416)
MQCPI035 ....using MQ protocol
MQCPI036 ....SSL Client side enabled with properties :
```

```

MQCPI139 .....secure socket protocols <NULL>
MQCPI031 .....cipher suites SSL_RSA_WITH_AES_256_CBC_SHA256
MQCPI032 .....key ring file C:\mqipt\samples\ssl\sslSample.pfx
MQCPI047 .....CA key ring file <NULL>
MQCPI071 .....site certificate uses
UID=*,CN=*,T=*,OU=*,DC=*,O=*,STREET=*,L=*,ST=*,PC=*,C=*,DNQ=*
MQCPI038 .....peer certificate uses
UID=*,CN=*,T=*,OU=*,DC=*,O=*,STREET=*,L=*,ST=*,PC=*,C=*,DNQ=*
MQCPI078 Route 1415 ready for connection requests

```

2. On the MQIPT 2 system:

a) Edit mqipt.conf and add the following route definition:

```

[route]
ListenerPort=1416
Destination=Server1.company2.com
DestinationPort=1414
SSLServer=true
SSLServerAskClientAuth=true
SSLServerKeyRing=C:\mqipt\samples\ssl\sslSample.pfx
SSLServerKeyRingPW=<mqiptPW>1!PCaB1HWrFMQp43ngjwgArg==!6N/vsbqru7iqMhFN+wozxQ==
SSLServerCipherSuites=SSL_RSA_WITH_AES_256_CBC_SHA256

```

b) Open a command prompt and start MQIPT:

```

C:
cd \mqipt\bin
mqipt .. -n ipt2

```

where .. indicates that the MQIPT configuration file, mqipt.conf, is in the parent directory, and ipt2 is the name to be given to the instance of MQIPT.

The following messages indicate that MQIPT has started successfully:

```

5724-H72 (C) Copyright IBM Corp. 2000, 2024. All Rights Reserved
MQCPI001 IBM MQ Internet Pass-Thru V9.4.0.0 starting
MQCPI004 Reading configuration information from mqipt.conf
MQCPI152 MQIPT name is ipt2
MQCPI021 Password checking has been enabled on the command port
MQCPI011 The path C:\mqipt\logs will be used to store the log files
MQCPI006 Route 1416 is starting and will forward messages to :
MQCPI034 ....Server1.company2.com(1414)
MQCPI035 ....using MQ protocol
MQCPI037 ....SSL Server side enabled with properties :
MQCPI139 .....secure socket protocols <NULL>
MQCPI031 .....cipher suites SSL_RSA_WITH_AES_256_CBC_SHA256
MQCPI032 .....key ring file C:\mqipt\samples\ssl\sslSample.pfx
MQCPI047 .....CA key ring file <NULL>
MQCPI071 .....site certificate uses
UID=*,CN=*,T=*,OU=*,DC=*,O=*,STREET=*,L=*,ST=*,PC=*,C=*,DNQ=*
MQCPI038 .....peer certificate uses
UID=*,CN=*,T=*,OU=*,DC=*,O=*,STREET=*,L=*,ST=*,PC=*,C=*,DNQ=*
MQCPI033 .....client authentication set to true
MQCPI078 Route 1416 ready for connection requests

```

3. At a command prompt on the IBM MQ client system, enter the following commands:

a) Set the **MQSERVER** environment variable:

```
SET MQSERVER=MQIPT.CONN.CHANNEL/tcp/10.9.1.2(1415)
```

b) Put a message:

```
amqsputc MQIPT.LOCAL.QUEUE MQIPT.QM1
Hello world
```

Press Enter twice after typing the message string.

c) Get the message:

```
amqsgetc MQIPT.LOCAL.QUEUE MQIPT.QM1
```

The message, "Hello world" is returned.

Authenticating a TLS client and server

You can run MQIPT as both a TLS server and client to terminate the incoming TLS session and forward data to the destination by using a separate TLS connection.

Before you begin

- Complete the steps that are listed in [“Getting started with IBM MQ Internet Pass-Thru”](#) on page 162.
- Read the topic [SSL/TLS support in MQIPT](#).

Note: This scenario uses self-signed certificates for convenience. Do not use any self-signed certificates in production environments. Instead, obtain certificates that are signed by a trusted certificate authority (CA).

About this task

The connection is made between an IBM MQ client and a IBM MQ server through a single instance of MQIPT. The connections between the IBM MQ client and MQIPT, and between MQIPT and the IBM MQ server, both use TLS. Therefore, the MQIPT route is both a TLS server and a TLS client.

During the TLS handshake between the client and MQIPT, the client and MQIPT send their certificates to each other to authenticate the connection. When the connection between the client and MQIPT is established, MQIPT establishes a separate TLS connection to the IBM MQ server. MQIPT and the IBM MQ server send their certificates to each other to authenticate the connection.

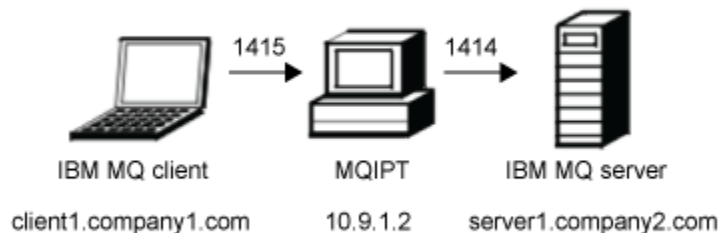


Figure 33. SSL/TLS server and client network diagram

This diagram shows the connection from the IBM MQ client (called client1.company1.com) through a single instance of MQIPT to the IBM MQ server (called server1.company2.com on port 1414).

Procedure

To configure an MQIPT route as both a TLS server and a TLS client, complete the following steps:

1. On the system where the client runs, complete the following steps to configure the IBM MQ client to use a TLS connection.

- a) Create a key repository for the client.

Enter the following command to create a new key repository named `clientkey.kdb` in the `C:\ProgramData\IBM\MQ` directory:

```
runmqakm -keydb -create -db C:\ProgramData\IBM\MQ\clientkey.kdb -pw password -stash
```

where `password` is the key repository password.

For more information, see [Setting up a key repository on AIX, Linux, and Windows](#).

- b) Create a personal certificate for the client, in the client key repository that you created in step “1.a” on page 173.

Enter the following command to create a new self-signed certificate with the label `clientcert` for the client:

```
runmqakm -cert -create -db C:\ProgramData\IBM\MQ\clientkey.kdb -stashed  
-label clientcert -dn CN=Client
```

For more information, see [Creating a self-signed personal certificate on AIX, Linux, and Windows](#).

- c) Extract the certificate from the client key repository.

Enter the following command to extract the client certificate:

```
runmqakm -cert -extract -db C:\ProgramData\IBM\MQ\clientkey.kdb -stashed -label clientcert  
-target C:\ProgramData\IBM\MQ\client.crt -format ascii
```

For more information, see [Extracting the public part of a self-signed certificate from a key repository on AIX, Linux, and Windows](#).

- d) Copy the extracted certificate file to the system where MQIPT runs.
2. On the system where the IBM MQ server runs, complete the following steps to configure the queue manager to use a TLS connection.

- a) Create a key repository for the queue manager.

Enter the following command to create a new key repository named `key.kdb` for the queue manager.

```
runmqakm -keydb -create -db C:\ProgramData\IBM\MQ\mqgrs\MQIPT!QM1\ssl\key.kdb -pw  
password -stash
```

where *password* is the key repository password.

Ensure that the mqm user is granted read access to the `C:\ProgramData\IBM\MQ\mqgrs\MQIPT!QM1\ssl\key.sth` stash file that the command creates to store the encrypted key repository password.

For more information, see [Setting up a key repository on AIX, Linux, and Windows](#).

- b) Create a personal certificate for the queue manager, in the queue manager key repository that you created in step “2.a” on page 174.

Enter the following command to create a new self-signed certificate with the label `ibmwebspheremqmqipt.qm1` for the queue manager:

```
runmqakm -cert -create -db C:\ProgramData\IBM\MQ\mqgrs\MQIPT!QM1\ssl\key.kdb -stashed  
-label ibmwebspheremqmqipt.qm1 -dn CN=MQIPT.QM1
```

For more information, see [Creating a self-signed personal certificate on AIX, Linux, and Windows](#).

- c) Extract the certificate from the queue manager key repository.

Enter the following command to extract the queue manager certificate:

```
runmqakm -cert -extract -db C:\ProgramData\IBM\MQ\mqgrs\MQIPT!QM1\ssl\key.kdb -stashed  
-label ibmwebspheremqmqipt.qm1  
-target C:\ProgramData\IBM\MQ\mqgrs\MQIPT!QM1\ssl\mqipt.qm1.crt -format ascii
```

For more information, see [Extracting the public part of a self-signed certificate from a key repository on AIX, Linux, and Windows](#).

- d) Copy the extracted certificate file to the system where MQIPT runs.
- e) Issue the following MQSC command to alter the MQIPT.CONN.CHANNEL server connection channel to use TLS:

```
ALTER CHANNEL(MQIPT.CONN.CHANNEL) CHLTYPE(SVRCONN) TRPTYPE(TCP)  
SSLCIPH(ANY_TLS12_OR_HIGHER)
```

3. On the system where MQIPT runs, complete the following steps to configure the MQIPT route to use TLS.

- a) Create a personal certificate for MQIPT in a PKCS #12 key repository.

V 9.4.0 **V 9.4.0** Enter the following command to create a new self-signed certificate with the label `mqipectert`:

```
mqipectKeytool -genkeypair -keystore C:\mqiptHome\ssl\mqipt.p12 -storetype pkcs12
-storepass password
      -alias mqipectert -dname "CN=MQIPT Test Certificate"
      -keyalg RSA -keysize 2048 -sigalg SHA256WithRSA
```

where *password* is the key repository password.

- b) Enter the following command to add the client certificate and the queue manager certificate to the MQIPT key repository:

V 9.4.0 **V 9.4.0**

```
mqipectKeytool -importcert -keystore C:\mqiptHome\ssl\mqipt.p12 -storetype pkcs12
-storepass password -file client.crt
mqiptKeytool -importcert -keystore C:\mqiptHome\ssl\mqipt.p12 -storetype pkcs12
-storepass password -file mqipt.qm1.crt
```

where *password* is the key repository password, *client.crt* is the client certificate file that you created in step “1.c” on page 174, and *mqipt.qm1.crt* is the queue manager certificate that you created in step “2.c” on page 174.

- c) Extract the MQIPT certificate from the key repository.

V 9.4.0 **V 9.4.0**

Enter the following command to extract the MQIPT certificate:

```
mqipectKeytool -exportcert -keystore C:\mqiptHome\ssl\mqipt.p12 -storetype pkcs12
-storepass password
      -alias mqipectert -file C:\mqiptHome\ssl\mqipt.crt -rfc
```

where *password* is the key repository password.

- d) Copy the extracted certificate file to both the system where the client runs and the system where the IBM MQ server runs.
- e) Enter the following command to encrypt the MQIPT key repository password:

```
mqiptPW
```

When prompted, enter the key repository password that you specified when you created the key repository in step “3.a” on page 174.

- f) Edit the `mqipt.conf` file and add the following route definition:

```
[route]
ListenerPort=1415
Destination=server1.company2.com
DestinationPort=1414
SSLServer=true
SSLServerKeyRing=C:\mqiptHome\ssl\mqipt.p12
SSLServerKeyRingPW=encrypted_password
SSLClient=true
SSLClientKeyRing=C:\mqiptHome\ssl\mqipt.p12
SSLClientKeyRingPW=encrypted_password
```

where *encrypted_password* is the encrypted key repository password created by running the **mqiptPW** command in step “3.e” on page 175.

4. Add the MQIPT certificate to both the client key repository and the queue manager key repository.

- a) On the system where the client runs, enter the following command to add the MQIPT certificate to the client key repository:

```
runmqakm -cert -add -db C:\ProgramData\IBM\MQ\clientkey.kdb -stashed
      -label mqipectert -file mqipt.crt -format ascii
```

where *mqipt.crt* is the MQIPT certificate file that you created in step “3.c” on page 175.

- b) On the system where the IBM MQ server runs, enter the following command to add the MQIPT certificate to the queue manager key repository:

```
runmqakm -cert -add -db C:\ProgramData\IBM\MQ\qmgrs\MQIPT!QM1\ssl\key.kdb -stashed  
-label mqiptcert -file mqipt.crt -format ascii
```

where *mqipt.crt* is the MQIPT certificate file that you created in step “3.c” on page 175.

For more information, see [Adding a CA certificate \(or the public part of a self-signed certificate\) into a key repository, on AIX, Linux, and Windows systems](#).

5. On the system where MQIPT runs, open a command prompt and enter the following commands to start MQIPT:

```
C:\mqipt\bin\mqipt C:\mqiptHome -n ipt1
```

where C:\mqiptHome indicates the location of the MQIPT configuration file, *mqipt.conf*, and *ipt1* is the name to be given to the instance of MQIPT.

The following messages indicate that MQIPT has started successfully:

```
5724-H72 (C) Copyright IBM Corp. 2000, 2024. All Rights Reserved  
MQCPI001 IBM MQ Internet Pass-Thru V9.4.0.0 starting  
MQCPI004 Reading configuration information from mqipt.conf  
MQCPI152 MQIPT name is ipt1  
MQCPI021 Password checking has been enabled on the command port  
MQCPI011 The path C:\mqiptHome\logs will be used to store the log files  
MQCPI006 Route 1415 is starting and will forward messages to :  
MQCPI034 ....server1.company2.com(1414)  
MQCPI035 ....using MQ protocol  
MQCPI036 ....SSL Client side enabled with properties :  
MQCPI139 .....secure socket protocols <NULL>  
MQCPI031 .....cipher suites <NULL>  
MQCPI032 .....key ring file C:\mqiptHome\ssl\mqipt.p12  
MQCPI047 .....CA key ring file <NULL>  
MQCPI071 .....site certificate uses  
UID=*,CN=*,T=*,OU=*,DC=*,O=*,STREET=*,L=*,ST=*,PC=*,C=*,DNQ=*,  
MQCPI038 .....peer certificate uses  
UID=*,CN=*,T=*,OU=*,DC=*,O=*,STREET=*,L=*,ST=*,PC=*,C=*,DNQ=*,  
MQCPI037 ....SSL Server side enabled with properties :  
MQCPI139 .....secure socket protocols <NULL>  
MQCPI031 .....cipher suites <NULL>  
MQCPI032 .....key ring file C:\mqiptHome\ssl\mqipt.p12  
MQCPI047 .....CA key ring file <NULL>  
MQCPI071 .....site certificate uses  
UID=*,CN=*,T=*,OU=*,DC=*,O=*,STREET=*,L=*,ST=*,PC=*,C=*,DNQ=*,  
MQCPI038 .....peer certificate uses  
UID=*,CN=*,T=*,OU=*,DC=*,O=*,STREET=*,L=*,ST=*,PC=*,C=*,DNQ=*,  
MQCPI033 .....client authentication set to false  
MQCPI078 Route 1415 ready for connection requests
```

6. At a command prompt on the IBM MQ client system, enter the following command to run the TLS sample program:

```
AMQSSSLC -m MQIPT.QM1 -c MQIPT.CONN.CHANNEL -x 10.9.1.2(1415)  
-k "C:\ProgramData\IBM\MQ\clientkey" -l clientcert -s ANY_TLS12_OR_HIGHER
```

The following message indicates that the application connected successfully to the queue manager:



```
Connection established to queue manager MQIPT.QM1
```

Configuring HTTP tunneling

In this scenario, you can test a simple connection between two instances of MQIPT over HTTP.

Before you begin

Before you start to use this scenario, make sure that you have completed the prerequisite tasks listed in [“Getting started with IBM MQ Internet Pass-Thru” on page 162](#).

Note:   From IBM MQ 9.4.0, MQIPT routes do not accept HTTP connections by default. Routes must be configured to accept HTTP connections by using the **AllowedProtocols** property.

About this task

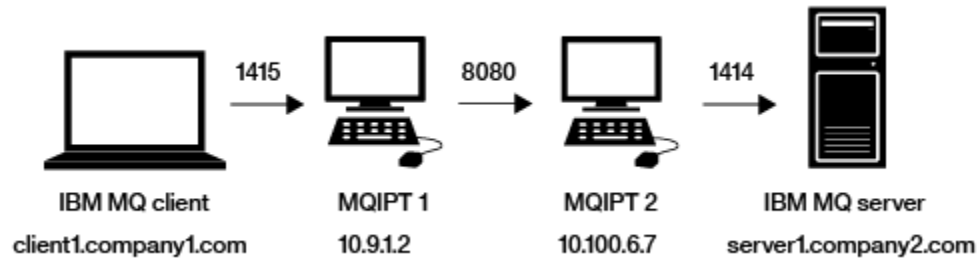


Figure 34. HTTP tunneling network diagram

This diagram shows the connection from the IBM MQ client (called client1.company1.com on port 1415) through two instances of MQIPT, tunnelling the connection over HTTP, and finally to the IBM MQ server (called server1.company2.com on port 1414).

Procedure

To configure HTTP tunneling between two instances of MQIPT, complete the following steps:

1. On the MQIPT 1 system:

a) Edit `mqipt.conf` and add the following route definition:

```
[route]
ListenerPort=1415
Destination=10.100.6.7
DestinationPort=8080
HTTP=true
HTTPServer=10.100.6.7
HTTPServerPort=8080
```

b) Open a command prompt and start MQIPT:

```
C:\mqipt\bin\mqipt C:\mqiptHome -n ipt1
```

where `C:\mqiptHome` indicates the location of the MQIPT configuration file, `mqipt.conf`, and `ipt1` is the name to be given to the instance of MQIPT.

The following messages indicate that MQIPT has started successfully:

```
5724-H72 (C) Copyright IBM Corp. 2000, 2024. All Rights Reserved
MQCPI001 IBM MQ Internet Pass-Thru V9.4.0.0 starting
MQCPI004 Reading configuration information from mqipt.conf
MQCPI152 MQIPT name is ipt1
MQCPI021 Password checking has been enabled on the command port
MQCPI011 The path C:\mqiptHome\logs will be used to store the log files
MQCPI006 Route 1415 is starting and will forward messages to :
MQCPI034 ....10.100.6.7(8080)
MQCPI035 ....using HTTP
MQCPI066 ....and HTTP server at 10.100.6.7(8080)
MQCPI078 Route 1415 ready for connection requests
```

2. On the MQIPT 2 system:

a) Edit `mqipt.conf` and add the following route definition:

```
[route]
ListenerPort=8080
Destination=Server1.company2.com
DestinationPort=1414
AllowedProtocols=http
```

b) Open a command prompt and start MQIPT:

```
C:\mqipt\bin\mqipt C:\mqiptHome -n ipt2
```

where C:\mqiptHome indicates the location of the MQIPT configuration file, mqipt.conf, and ipt2 is the name to be given to the instance of MQIPT.

The following messages indicate that MQIPT has started successfully:

```
5724-H72 (C) Copyright IBM Corp. 2000, 2024. All Rights Reserved
MQCPI001 IBM MQ Internet Pass-Thru V9.4.0.0 starting
MQCPI004 Reading configuration information from mqipt.conf
MQCPI152 MQIPT name is ipt2
MQCPI021 Password checking has been enabled on the command port
MQCPI011 The path C:\mqiptHome\logs will be used to store the log files
MQCPI006 Route 8080 is starting and will forward messages to :
MQCPI034 ....Server1.company2.com(1414)
MQCPI035 ....using MQ protocols
MQCPI158 ....connection protocols accepted: HTTP
MQCPI078 Route 8080 ready for connection requests
```

3. At a command prompt on the IBM MQ client, enter the following commands:

a) Set the **MQSERVER** environment variable:

```
SET MQSERVER=MQIPT.CONN.CHANNEL/tcp/10.9.1.2(1415)
```

b) Put a message:

```
amqsputc MQIPT.LOCAL.QUEUE MQIPT.QM1
Hello world
```

Press Enter twice after typing the message string.

c) Get the message:

```
amqsgetc MQIPT.LOCAL.QUEUE MQIPT.QM1
```

The message, "Hello world" is returned.

Configuring access control

In this scenario, you can set up your MQIPT to only accept connections from specific clients by using the Java security manager to add security checks on the MQIPT listener port.

Before you begin

- Before you start to use this scenario, make sure that you have completed the prerequisite tasks listed in [“Getting started with IBM MQ Internet Pass-Thru”](#) on page 162.

About this task

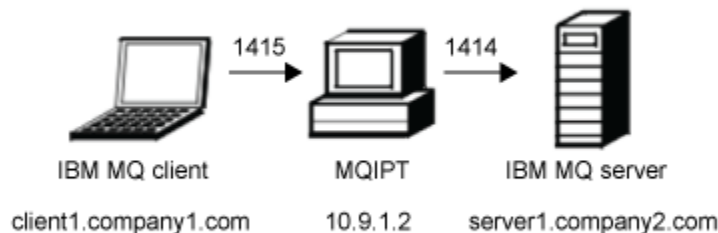


Figure 35. Access control network diagram

This diagram shows the connection from the IBM MQ client (called client1.company1.com on port 1415) through MQIPT to the IBM MQ server (called server1.company2.com on port 1414).

Procedure

To configure access control, complete the following steps:

1. Set up MQIPT:

- a) Copy the sample Java security manager policy to the MQIPT home directory by entering the following command at a command prompt:

```
copy C:\mqipt\samples\mqiptSample.policy C:\mqiptHome\mqipt.policy
```

- b) Start the Policy Tool utility by using the following command:

```
C:\mqipt\java\jre\bin\policytool
```

- c) Click **File > Open** then select C:\mqiptHome\mqipt.policy..

- d) Click **Edit Policy Entry** then change CodeBase from:

```
file:/C:/Program Files/IBM/IBM MQ Internet Pass-Thru/lib/com.ibm.mq.ipt.jar
```

to:

```
file:/C:/mqipt/lib/com.ibm.mq.ipt.jar
```

- e) Change the file permissions for the IBM MQ Internet Pass-Thru, errors and logs directories from:

```
C:\Program Files\IBM\IBM MQ Internet Pass-Thru
```

to:

```
C:\mqiptHome
```

- f) Change the other file permissions from:

```
C:\Program Files\IBM\IBM MQ Internet Pass-Thru
```

to:

```
C:\mqipt
```

- g) Click **Add Permission**

Complete the fields as follows:

Permission: java.net.SocketPermission

Target: client1.company1.com:1024-

Actions: accept, listen, resolve

- h) Click **File > Save** to save the changes to the policy file.

- i) Edit mqipt.conf.

- i) Add the following two properties to the [global] section:

```
SecurityManager=true  
SecurityManagerPolicy=C:\mqiptHome\mqipt.policy
```

- ii) Add the following route definition:

```
[route]  
ListenerPort=1415  
Destination=server1.company2.com  
DestinationPort=1414
```

2. Start MQIPT:

Open a command prompt and enter the following:

```
C:\mqipt\bin\mqipt C:\mqiptHome -n ipt1
```

where C:\mqiptHome indicates the location of the MQIPT configuration file, mqipt.conf, and ipt1 is the name to be given to the instance of MQIPT.

The following messages indicate that MQIPT has started successfully:

```
5724-H72 (C) Copyright IBM Corp. 2000, 2024. All Rights Reserved
MQCPI001 IBM MQ Internet Pass-Thru V9.4.0.0 starting
MQCPI004 Reading configuration information from mqipt.conf
MQCPI152 MQIPT name is ipt1
MQCPI055 Setting the java.security.policy to C:\mqiptHome\mqipt.policy
MQCPI053 Starting the Java Security Manager
MQCPI021 Password checking has been enabled on the command port
MQCPI011 The path C:\mqiptHome\logs will be used to store the log files
MQCPI006 Route 1415 has started and will forward messages to :
MQCPI034 ....server1.company2.com(1414)
MQCPI035 ....using MQ protocol
MQCPI078 Route 1415 ready for connection requests
```

3. At a command prompt on the IBM MQ client system, enter the following commands:

a) Set the **MQSERVER** environment variable:

```
SET MQSERVER=MQIPT.CONN.CHANNEL/tcp/10.9.1.2(1415)
```

b) Put a message:

```
amqsputc MQIPT.LOCAL.QUEUE MQIPT.QM1
Hello world
```

Press Enter twice after typing the message string.

c) Get the message:

```
amqsgetc MQIPT.LOCAL.QUEUE MQIPT.QM1
```

The message, "Hello world" is returned.

Configuring a SOCKS proxy

In this scenario, you can make MQIPT act as a SOCKS proxy.

Before you begin

- Before you start to use this scenario, make sure that you have completed the prerequisite tasks listed in [“Getting started with IBM MQ Internet Pass-Thru” on page 162](#).
- Enable SOCKS on either the whole IBM MQ computer or just the IBM MQ client applications **amqsputc** and **amqsgetc**.
- Configure the SOCKS client as follows:
 1. Use MQIPT as the SOCKS proxy.
 2. Enable SOCKS 5 support.
 3. Disable user authentication.
 4. Restrict connections to the MQIPT network address.

About this task

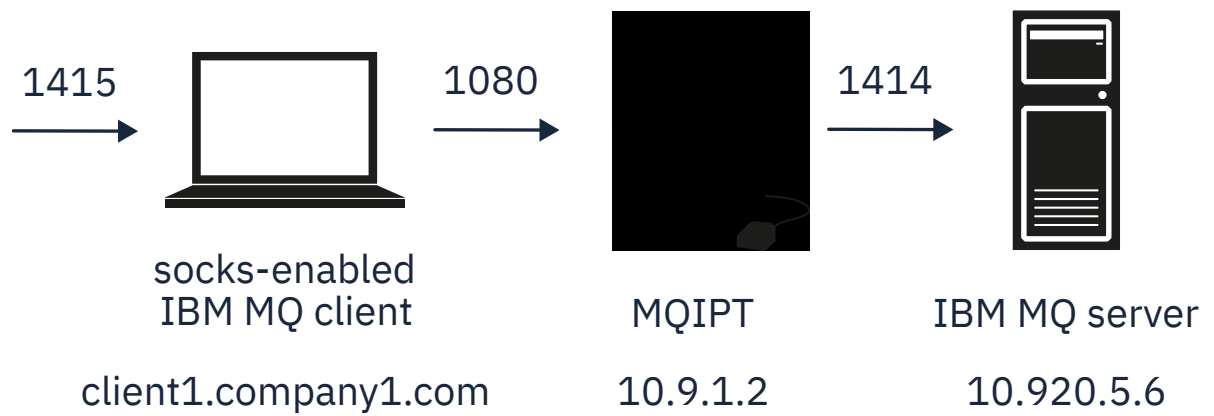


Figure 36. SOCKS proxy network diagram

This diagram shows the connection flow from the IBM MQ client (called client1.company1.com on port 1415) through MQIPT to the IBM MQ server (called server1.company2.com on port 1414).

Procedure

To configure a SOCKS proxy, complete the following steps:

1. Configure and start MQIPT:

a) Edit `mqipt.conf` and add the following route definition:

```
[route]
ListenerPort=1080
Destination=server1.company2.com
DestinationPort=1414
SocksServer=true
```

The values of the **Destination** and **DestinationPort** route properties are ignored as the true destination is obtained from the IBM MQ client during the SOCKS handshaking process.

b) Open a command prompt and start MQIPT:

```
C:\mqipt\bin\mqipt C:\mqiptHome -n ipt1
```

where `C:\mqiptHome` indicates the location of the MQIPT configuration file, `mqipt.conf`, and `ipt1` is the name to be given to the instance of MQIPT.

The following messages indicate that MQIPT has started successfully:

```
5724-H72 (C) Copyright IBM Corp. 2000, 2024. All Rights Reserved
MQCPI001 IBM MQ Internet Pass-Thru V9.4.0.0 starting
MQCPI004 Reading configuration information from mqipt.conf
MQCPI152 MQIPT name is ipt1
MQCPI021 Password checking has been enabled on the command port
MQCPI011 The path C:\mqiptHome\logs will be used to store the log files
MQCPI006 Route 1080 has started and will forward messages to :
MQCPI034 ....server1.company2.com(1414)
MQCPI035 ....using MQ protocol
MQCPI052 ....SOCKS server side enabled
MQCPI078 Route 1080 ready for connection requests
```

2. At a command prompt on the IBM MQ client system, enter the following commands:

a) Set the **MQSERVER** environment variable:

```
SET MQSERVER=MQIPT.CONN.CHANNEL/tcp/10.20.5.6(1414)
```

b) Put a message:

```
amqsputc MQIPT.LOCAL.QUEUE MQIPT.QM1
Hello world
```

Press Enter twice after typing the message string.

c) Get the message:

```
amqsgetc MQIPT.LOCAL.QUEUE MQIPT.QM1
```

The message, "Hello world" is returned.

Configuring a SOCKS client

In this scenario, you can run MQIPT as though it was SOCKS-enabled, using an existing SOCKS proxy.

This is similar to the scenario [“Configuring a SOCKS proxy” on page 180](#), except that MQIPT makes a SOCKS-enabled connection instead of the IBM MQ client.

Before you begin

Before you start to use this scenario, make sure that you have completed the prerequisite tasks listed in [“Getting started with IBM MQ Internet Pass-Thru”](#) on page 162

About this task

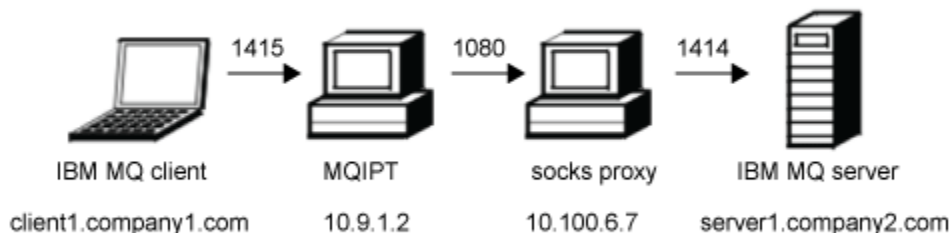


Figure 37. SOCKS client network diagram

This diagram shows the network connection from the IBM MQ client (called client1.company1.com on port 1415) through MQIPT, then through the SOCKS proxy (on port 1080) to the IBM MQ server (called server1.company2.com on port 1414).

Procedure

To configure a SOCKS client, complete the following steps:

1. Set up MQIPT.

On the MQIPT computer, edit `mqipt.conf` and add a route definition:

```
[route]
ListenerPort=1415
Destination=server1.company2.com
DestinationPort=1414
SocksClient=true
SocksProxyHost=10.9.6.7
SocksProxyPort=1080
```

2. Start MQIPT.

Open a command prompt and enter:

```
C:\mqipt\bin\mqipt C:\mqiptHome -n ipt1
```

where `C:\mqiptHome` indicates the location of the MQIPT configuration file, `mqipt.conf`, and `ipt1` is the name to be given to the instance of MQIPT.

The following messages indicate that MQIPT has started successfully:

```
5724-H72 (C) Copyright IBM Corp. 2000, 2024. All Rights Reserved
MQCPI001 IBM MQ Internet Pass-Thru V9.4.0.0 starting
MQCPI004 Reading configuration information from mqipt.conf
MQCPI152 MQIPT name is ipt1
MQCPI021 Password checking has been enabled on the command port
MQCPI011 The path C:\mqiptHome\logs will be used to store the log files
MQCPI006 Route 1415 has started and will forward messages to :
MQCPI034 ....server1.company2.com(1414)
MQCPI035 ....using MQ protocol
MQCPI039 ....and SOCKS proxy at 10.9.6.7(1080)
MQCPI078 Route 1415 ready for connection requests
```

3. At a command prompt on the IBM MQ client, enter the following commands:

- a) Set the **MQSERVER** environment variable:

```
SET MQSERVER=MQIPT.CONN.CHANNEL/tcp/10.9.1.2(1415)
```

b) Put a message:

```
amqsputc MQIPT.LOCAL.QUEUE MQIPT.QM1  
Hello world
```

Press Enter twice after typing the message string.

c) Get the message:

```
amqsgetc MQIPT.LOCAL.QUEUE MQIPT.QM1
```

The message, "Hello world" is returned.

Configuring MQIPT clustering support

In this scenario, you can set up a clustering environment.

Before you begin

- Before you start to use this scenario, make sure that you have completed the prerequisite tasks listed in [“Getting started with IBM MQ Internet Pass-Thru”](#) on page 162.
- On the IBM MQ server LONDON:
 - Defined a queue manager called LONDON.
 - Defined a server connection channel called MQIPT.CONN.CHANNEL.
 - Started a TCP/IP listener for LONDON on port 1414.
 - SOCKS-enabled the queue manager.
- On the IBM MQ server NEWYORK:
 - Defined a queue manager called NEWYORK.
 - Defined a server connection channel called MQIPT.CONN.CHANNEL.
 - Started a TCP/IP listener for NEWYORK on port 1414.
 - SOCKS-enabled the queue manager.

Note: To SOCKS-enable a queue manager, enable either the whole computer or just the IBM MQ server application. Configure the SOCKS client as follows:

- Point the client to MQIPT as the SOCKS proxy.
- Enable SOCKS V5 support.
- Disable user authentication.
- Only make remote connections to the MQIPT.

About this task

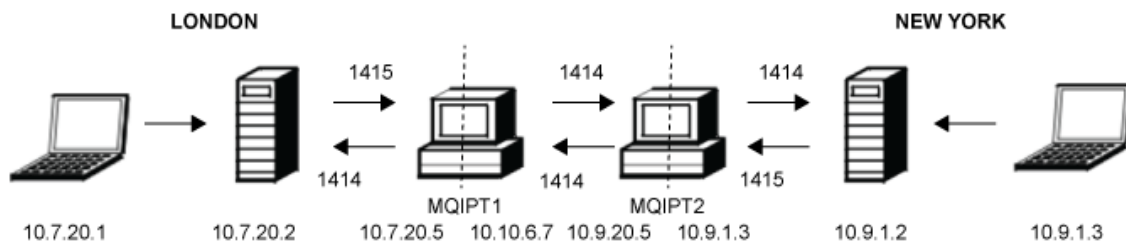


Figure 38. Clustering network diagram

This diagram shows the connections from the IBM MQ clients through MQIPT to the IBM MQ servers.

Only one application can listen on a given port on the same computer. If port 1414 is already in use, choose a free port and substitute it in the examples.

You can then test the routes between the queue managers by putting a message on the local queue on the LONDON server and retrieving it from the NEWYORK server.

Procedure

To configure MQIPT clustering support, complete the following steps:

1. Set up the LONDON server.

Open a command prompt and enter the following commands:

```
runmqsc
DEFINE CHANNEL(TO.LONDON) +
  CHLTYPE(CLUSRCVR) TRPTYPE(TCP) +
  CLUSTER(INVENTORY) +
  CONNAME('10.10.6.7(1414)')
DEFINE CHANNEL(TO.NEWYORK) +
  CHLTYPE(CLUSSDR) TRPTYPE(TCP) +
  CLUSTER(INVENTORY) +
  CONNAME('10.9.20.5(1414)')
```

2. Set up the NEWYORK server

Open a command prompt and enter the following commands:

```
runmqsc
ALTER QMGR REPOS(INVENTORY)
DEFINE QLOCAL(MQIPT.LOCAL.QUEUE) +
  CLUSTER(INVENTORY)
DEFINE CHANNEL(TO.NEWYORK) +
  CHLTYPE(CLUSRCVR) TRPTYPE(TCP) +
  CLUSTER(INVENTORY) +
  CONNAME('10.9.20.5(1414)')
DEFINE CHANNEL(TO.LONDON) +
  CHLTYPE(CLUSSDR) TRPTYPE(TCP) +
  CLUSTER(INVENTORY) +
  CONNAME('10.10.6.7(1414)')
```

3. Set up MQIPT 1.

Edit `mqipt.conf` and add the following route definitions:

```
[route]
Name=LONDON to NEWYORK
ListenerPort=1415
Destination=10.9.20.5
DestinationPort=1414
SocksServer=true

[route]
Name=MQIPT1 to LONDON
ListenerPort=1414
Destination=10.7.20.2
DestinationPort=1414
```

4. Start MQIPT 1.

Open a command prompt and enter:

```
C:\mqipt\bin\mqipt C:\mqiptHome -n ipt1
```

where `C:\mqiptHome` indicates the location of the MQIPT configuration file, `mqipt.conf`, and `ipt1` is the name to be given to the instance of MQIPT.

The following messages indicate that MQIPT has started successfully:

```
5724-H72 (C) Copyright IBM Corp. 2000, 2024. All Rights Reserved
MQCPI001 IBM MQ Internet Pass-Thru V9.4.0.0 starting
MQCPI004 Reading configuration information from mqipt.conf
MQCPI152 MQIPT name is ipt1
MQCPI021 Password checking has been enabled on the command port
```

```

MQCPI011 The path C:\mqiptHome\logs will be used to store the log files
MQCPI006 Route 1415 has started and will forward messages to :
MQCPI034 ....10.9.20.5(1414)
MQCPI035 ....using MQ protocol
MQCPI052 ....SOCKS server side enabled
MQCPI078 Route 1415 ready for connection requests
MQCPI006 Route 1414 has started and will forward messages to :
MQCPI034 ....10.7.20.2(1414)
MQCPI035 ....using MQ protocol
MQCPI078 Route 1414 ready for connection requests

```

5. Set up MQIPT 2.

Edit `mqipt.conf` and add the following route definitions:

```

[route]
Name=NEWYORK to LONDON
ListenerPort=1415
Destination=10.10.6.7
DestinationPort=1414
SocksServer=true

[route]
Name=MQIPT2 to NEWYORK
ListenerPort=1414
Destination=10.9.1.2
DestinationPort=1414

```

6. Start MQIPT 2.

Open a command prompt and enter the following commands:

```

C:
cd \mqipt\bin
mqipt .. -n ipt2

```

where `..` indicates that the MQIPT configuration file, `mqipt.conf`, is in the parent directory, and `ipt2` is the name to be given to the instance of MQIPT.

The following messages indicate that MQIPT has started successfully:

```

5724-H72 (C) Copyright IBM Corp. 2000, 2024. All Rights Reserved
MQCPI001 IBM MQ Internet Pass-Thru V9.4.0.0 starting
MQCPI004 Reading configuration information from mqipt.conf
MQCPI152 MQIPT name is ipt2
MQCPI021 Password checking has been enabled on the command port
MQCPI011 The path C:\mqipt\logs will be used to store the log files
MQCPI006 Route 1415 has started and will forward messages to :
MQCPI034 ....10.10.6.7(1414)
MQCPI035 ....using MQ protocol
MQCPI052 ....SOCKS server side enabled
MQCPI078 Route 1415 ready for connection requests
MQCPI006 Route 1414 has started and will forward messages to :
MQCPI034 ....10.9.1.2(1414)
MQCPI035 ....using MQ protocol
MQCPI078 Route 1414 ready for connection requests

```

7. At a command prompt on the LONDON IBM MQ client (10.7.20.1), enter the following commands:

a) Set the **MQSERVER** environment variable:

```
SET MQSERVER=MQIPT.CONN.CHANNEL/tcp/10.7.20.2(1414)
```

b) Put a message:

```
amqsputc MQIPT.LOCAL.QUEUE LONDON
Hello world
```

Press Enter twice after typing the message string.

This causes the LONDON queue manager to send messages to the queue on the NEW YORK queue manager.

8. At a command prompt on the NEW YORK IBM MQ client (10.9.1.3), enter the following commands:

a) Set the **MQSERVER** environment variable:

```
SET MQSERVER=MQIPT.CONN.CHANNEL/TCP/10.9.1.2(1414)
```

b) Get the message:

```
amqsgetc MQIPT.LOCAL.QUEUE NEWYORK
```

The message, "Hello world" is returned.

Allocating port numbers

You can control the local port addresses used when making outgoing connections. For example, if your firewall allows only certain ranges of port numbers, you can use MQIPT to ensure that output originates from a valid port.

Before you begin

- Before you start to use this scenario, make sure that you have completed the prerequisite tasks listed in [“Getting started with IBM MQ Internet Pass-Thru”](#) on page 162.
- Install MQIPT on a multihomed computer.

About this task

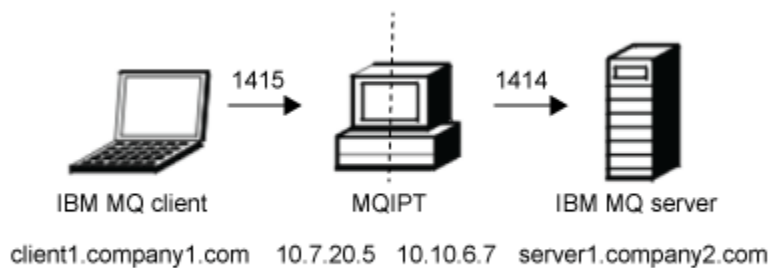


Figure 39. Port allocation network diagram

This diagram shows the connection from an IBM MQ client (client1.company1.com on port 1415) through MQIPT to an IBM MQ server (server1.company2.com on port 1414).

Procedure

To allocate port numbers, complete the following steps:

1. Set up MQIPT.

Edit `mqipt.conf` and add the following route definition:

```
[route]
ListenerPort=1415
Destination=server1.company2.com
DestinationPort=1414
LocalAddress=10.10.6.7
OutgoingPort=2000
MaxConnectionThreads=20
```

2. Start MQIPT.

Open a command prompt on the IBM MQ system, and enter the following command:

```
C:\mqipt\bin\mqipt C:\mqiptHome -n ipt1
```

where `C:\mqiptHome` indicates the location of the MQIPT configuration file, `mqipt.conf`, and `ipt1` is the name to be given to the instance of MQIPT.

The following messages indicate that MQIPT has started successfully:

```
5724-H72 (C) Copyright IBM Corp. 2000, 2024. All Rights Reserved
MQCPI001 IBM MQ Internet Pass-Thru V9.4.0.0 starting
MQCPI004 Reading configuration information from mqipt.conf
MQCPI152 MQIPT name is ipt1
MQCPI021 Password checking has been enabled on the command port
MQCPI011 The path C:\mqiptHome\logs will be used to store the log files
MQCPI006 Route 1415 is starting and will forward messages to :
MQCPI034 ....server1.company2.com(1414)
MQCPI035 ....using MQ protocol
MQCPI069 ....binding to local address 10.10.6.7 when making new connections
MQCPI070 ....using local port address range 2000-2019 when making new connections
MQCPI078 Route 1415 ready for connection requests
```

3. At a command prompt on the IBM MQ client system, enter the following commands:

a) Set the **MQSERVER** environment variable:

```
SET MQSERVER=MQIPT.CONN.CHANNEL/tcp/10.7.20.5(1415)
```

b) Put a message:

```
amqsputc MQIPT.LOCAL.QUEUE MQIPT.QM1
Hello world
```

Press Enter twice after typing the message string.

c) Get the message:

```
amqsgetc MQIPT.LOCAL.QUEUE MQIPT.QM1
```

The message, "Hello world" is returned.

Retrieving CRLs by using an LDAP server

You can configure MQIPT to use an LDAP server to retrieve certificate revocation lists (CRLs).

Before you begin

- Before you start to use this scenario, make sure that you have completed the prerequisite tasks listed in [“Getting started with IBM MQ Internet Pass-Thru” on page 162](#).
- Ensure that MQIPT 2 has a personal certificate, issued by the trusted Certificate Authority (CA), stored in a key ring file called `myCert.pfx`.
- Ensure that MQIPT 1 has a copy of the trusted CA certificate that will be used to authenticate the certificate sent by MQIPT 2. This certificate is stored in a key ring file called `caCerts.pfx`.
- The passwords to access the key rings have been encrypted using the **mqiptPW** command.

About this task

In this scenario, you can connect the IBM MQ client to a queue manager (QM) and place an IBM MQ message on the target queue. Running an MQIPT trace on MQIPT 1 will show the LDAP server being used.

To demonstrate how CRLs work, make sure that the personal certificate used by MQIPT 2 is revoked by the trusted CA. Then the IBM MQ client is not allowed to connect to the QM, as the connection from MQIPT 1 to MQIPT 2 is rejected.

It is not the intention of this scenario to explain how to install and set up an LDAP server nor how to create a key ring file containing personal or trusted certificates. It assumes that the LDAP server is available from a known and trusted CA. A backup LDAP server is not used, but could be implemented by adding the appropriate Route properties.

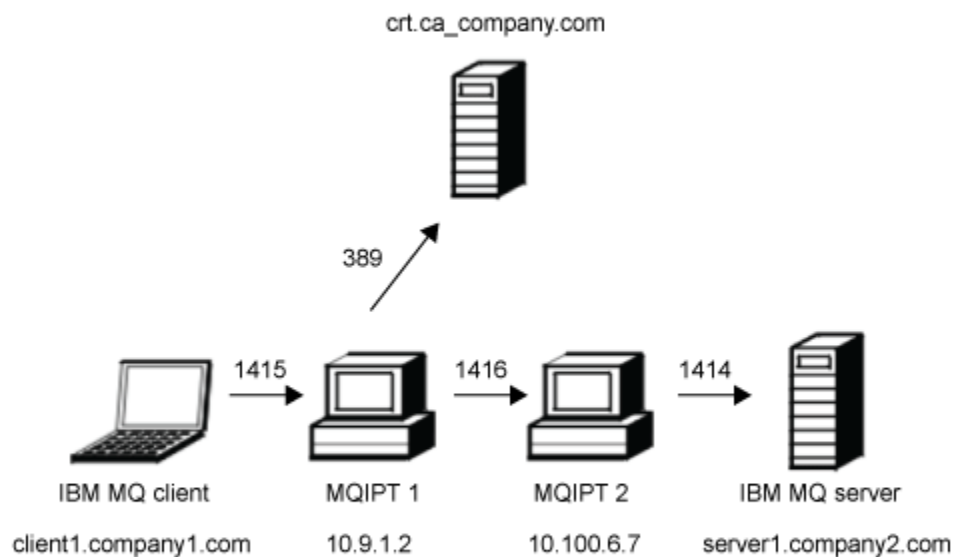


Figure 40. LDAP server network diagram

This diagram shows the connection from the IBM MQ client (`client1.company1.com` on port 1415) through two instances of MQIPT to the IBM MQ server (`server1.company2.com` on port 1414). The first MQIPT has a connection to an LDAP server (`crl.ca_company.com` on port 389).

Procedure

To retrieve CRLs by using an LDAP server, complete the following steps:

1. On the MQIPT 1 system:

a) Edit `mqipt.conf` and add the following route definition:

```
[route]
ListenerPort=1415
Destination=10.100.6.7
DestinationPort=1416
SSLClient=true
SSLClientCAKeyRing=C:\mqiptHome\ssl\caCerts.pfx
SSLClientCAKeyRingPW=encrypted_key_ring_password
LDAP=true
LDAPServer1=crl.ca_company.com
LDAPServer1Timeout=4
```

where `encrypted_key_ring_password` is the password for the `caCerts.pfx` key ring, encrypted using the **mqiptPW** command.

b) Open a command prompt and start MQIPT:

```
C:\mqipt\bin\mqipt C:\mqiptHome -n ipt1
```

where `C:\mqiptHome` indicates the location of the MQIPT configuration file, `mqipt.conf`, and `ipt1` is the name to be given to the instance of MQIPT.

The following messages indicate that MQIPT has started successfully:

```
5724-H72 (C) Copyright IBM Corp. 2000, 2024. All Rights Reserved
MQCPI001 IBM MQ Internet Pass-Thru V9.4.0.0 starting
```

```

MQCPI004 Reading configuration information from mqipt.conf
MQCPI152 MQIPT name is ipt1
MQCPI021 Password checking has been enabled on the command port
MQCPI011 The path C:\mqiptHome\logs will be used to store the log files
MQCPI006 Route 1415 has started and will forward messages to :
MQCPI034 ....10.100.6.7(1416)
MQCPI035 ....using MQ protocol
MQCPI036 ....SSL Client side enabled with properties :
MQCPI031 .....CipherSuites <NULL>
MQCPI032 .....key ring file <NULL>
MQCPI047 .....CA key ring file C:\mqiptHome\ssl\caCerts.pfx
MQCPI071 .....site certificate uses UID=*,CN=*,T=*,OU=*,DC=*,O=*,
                                STREET=*,L=*,ST=*,PC=*,C=*,DNQ=*
MQCPI038 .....peer certificate uses UID=*,CN=*,T=*,OU=*,DC=*,O=*,
                                STREET=*,L=*,ST=*,PC=*,C=*,DNQ=*
MQCPI075 ....LDAP main server at crl.ca_company.com(389)
MQCPI086 .....timeout of 4 second(s)
MQCPI084 ....CRL cache expiry timeout is 1 hour(s)
MQCPI085 ....CRLs will be saved in the key-ring file(s)
MQCPI078 Route 1415 ready for connection requests

```

2. On the MQIPT 2 system:

- a) Edit `mqipt.conf` and add the following route definition:

```

[route]
ListenerPort=1416
Destination=server1.company2.com
DestinationPort=1414
SSLServer=true
SSLServerKeyRing=C:\mqipt\ssl\myCert.pfx
SSLServerKeyRingPW=encrypted_key_ring_password

```

where `encrypted_key_ring_password` is the password for the `myCert.pfx` key ring, encrypted using the **mqiptPW** command.

- b) Open a command prompt and start MQIPT:

```

C:
cd \mqipt\bin
mqipt .. -n ipt2

```

where `..` indicates that the MQIPT configuration file, `mqipt.conf`, is in the parent directory, and `ipt2` is the name to be given to the instance of MQIPT.

The following message indicates successful completion:

```

5724-H72 (C) Copyright IBM Corp. 2000, 2024. All Rights Reserved
MQCPI001 IBM MQ Internet Pass-Thru V9.4.0.0 starting
MQCPI004 Reading configuration information from mqipt.conf
MQCPI152 MQIPT name is ipt2
MQCPI021 Password checking has been enabled on the command port
MQCPI011 The path C:\mqipt\logs will be used to store the log files
MQCPI006 Route 1416 is starting and will forward messages to :
MQCPI034 ....server1.company2.com(1414)
MQCPI035 ....using MQ protocol
MQCPI037 ....SSL Server side enabled with properties :
MQCPI031 .....CipherSuites <NULL>
MQCPI032 .....key ring file C:\mqipt\ssl\myCert.pfx
MQCPI047 .....CA key ring file <NULL>
MQCPI071 .....site certificate uses UID=*,CN=*,T=*,OU=*,DC=*,O=*,
                                STREET=*,L=*,ST=*,PC=*,C=*,DNQ=*
MQCPI038 .....peer certificate uses UID=*,CN=*,T=*,OU=*,DC=*,O=*,
                                STREET=*,L=*,ST=*,PC=*,C=*,DNQ=*
MQCPI033 .....client authentication set to false
MQCPI078 Route 1416 ready for connection requests

```

3. At a command prompt on the IBM MQ client system, enter the following commands:

- a) Set the **MQSERVER** environment variable:

```

SET MQSERVER=MQIPT.CONN.CHANNEL/tcp/10.9.1.2(1415)

```

- b) Put a message:

```
amqspu tc MQIPT.LOCAL.QUEUE MQIPT.QM1
Hello world
```

Press Enter twice after typing the message string.

c) Get the message:

```
amqsgetc MQIPT.LOCAL.QUEUE MQIPT.QM1
```

The message, "Hello world" is returned.

Running MQIPT in TLS proxy mode

You can run MQIPT in TLS proxy mode, so that it accepts a TLS connection request from an IBM MQ TLS client and tunnels it to an IBM MQ TLS server.

Before you begin

Before you start to use this scenario, make sure that you have completed the prerequisite tasks listed in [“Getting started with IBM MQ Internet Pass-Thru”](#) on page 162.

About this task

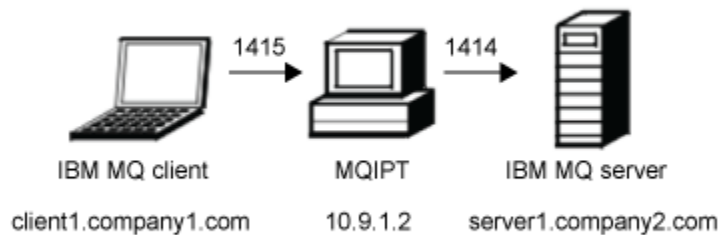


Figure 41. SSL/TLS proxy mode network diagram

This diagram shows the connection flow from the IBM MQ client (client1.company1.com on port 1415) through MQIPT to the IBM MQ server (server1.company2.com on port 1414).

For further information on configuring TLS for IBM MQ, refer to [Working with SSL/TLS](#).

Procedure

To run MQIPT in TLS proxy mode, complete the following steps:

1. Configure the IBM MQ client and server to use a TLS connection.
 - a) Create a key repository for the queue manager.
For more information, see [Setting up a key repository on AIX, Linux, and Windows](#).
 - b) Create a key repository for the client in the C:\ProgramData\IBM\MQ directory. Call it *clientkey.kdb*.
 - c) Create a personal certificate for the queue manager, in the queue manager key repository that you created in step “1.a” on page 191.
For more information, see [Creating a self-signed personal certificate on AIX, Linux, and Windows](#).
 - d) Create a personal certificate for the client, in the client key repository that you created in step “1.b” on page 191.
 - e) Extract the personal certificate from the server key repository and add it to the client repository.

For more information, see [Extracting the public part of a self-signed certificate from a key repository on AIX, Linux, and Windows](#), and [Adding a CA certificate \(or the public part of a self-signed certificate\) into a key repository, on AIX, Linux, and Windows systems](#).

- f) Extract the personal certificate from the client key repository and add it to the server key repository.
- g) Alter the MQIPT.CONN.CHANNEL server connection channel to use TLS by using the MQSC command:

```
ALTER CHANNEL(MQIPT.CONN.CHANNEL) CHLTYPE(SVRCONN) TRPTYPE(TCP)
SSLCPH(TLS_RSA_WITH_AES_128_CBC_SHA256)
```

2. To run MQIPT in TLS proxy mode, complete the following steps:

- a) Edit `mqipt.conf` and add the following route definition:

```
[route]
ListenerPort=1415
Destination=server1.company2.com
DestinationPort=1414
SSLProxyMode=true
```

- b) Start MQIPT.

Open a command prompt and enter the following command:

```
C:\mqipt\bin\mqipt C:\mqiptHome -n ipt1
```

where `C:\mqiptHome` indicates the location of the MQIPT configuration file, `mqipt.conf`, and `ipt1` is the name to be given to the instance of MQIPT.

The following messages indicate that MQIPT has started successfully:

```
5724-H72 (C) Copyright IBM Corp. 2000, 2024. All Rights Reserved
MQCPI001 IBM MQ Internet Pass-Thru V9.4.0.0 starting
MQCPI004 Reading configuration information from mqipt.conf
MQCPI152 MQIPT name is ipt1
MQCPI021 Password checking has been enabled on the command port
MQCPI011 The path C:\mqiptHome\logs will be used to store the log files
MQCPI006 Route 1415 has started and will forward messages to :
MQCPI034 ...server1.company2.com(1414)
MQCPI035 ...using SSLProxyMode protocol
MQCPI078 Route 1415 ready for connection requests
```

3. At a command prompt on the IBM MQ client system, enter the following command to run the TLS sample program:


```
AMQSSSLC -m MQIPT.QM1 -c MQIPT.CONN.CHANNEL -x 10.9.1.2(1415)
-k "C:\ProgramData\IBM\MQ\clientkey" -l cert_label -s
TLS_RSA_WITH_AES_128_CBC_SHA256
```

where `cert_label` is the label of the client certificate that you created in step “1.d” on [page 191](#).

Running MQIPT in TLS proxy mode with a security manager

You can run MQIPT in TLS proxy mode, so that it accepts a TLS connection request from an IBM MQ TLS client and tunnels it to an IBM MQ TLS server. By using a security manager with MQIPT, you can restrict the addresses to which messages can be sent.

Before you begin

Note:  The use of the Java security manager with MQIPT is deprecated due to the Java security manager having been deprecated for removal in a future release of Java.

Before you start to use this scenario, make sure that you have completed the prerequisite tasks listed in [“Getting started with IBM MQ Internet Pass-Thru” on page 162](#).

About this task

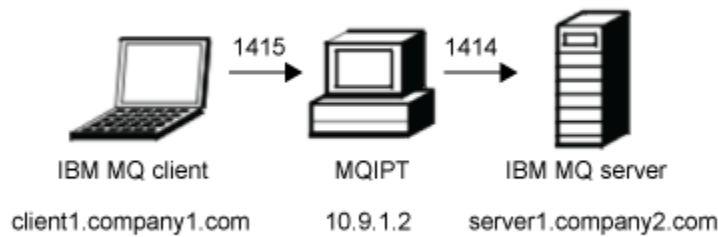


Figure 42. SSL/TLS proxy mode network diagram

This diagram shows the connection flow from the IBM MQ client (`client1.company1.com` on port 1415) through MQIPT to the IBM MQ server (`server1.company2.com` on port 1414).

For further information on configuring TLS for IBM MQ, refer to [Working with SSL/TLS](#).

Procedure

To run MQIPT in TLS proxy mode with a security manager, complete the following steps:

1. Configure the IBM MQ client and server to use a TLS connection.
 - a) Create a key repository for the queue manager.
For more information, see [Setting up a key repository on AIX, Linux, and Windows](#).
 - b) Create a key repository for the client in the `C:\ProgramData\IBM\MQ` directory. Call it `clientkey.kdb`.
 - c) Create a personal certificate for the queue manager, in the queue manager key repository that you created in step “1.a” on page 193.
For more information, see [Creating a self-signed personal certificate on AIX, Linux, and Windows](#).
 - d) Create a personal certificate for the client, in the client key repository that you created in step “1.b” on page 193.
 - e) Extract the personal certificate from the server key repository and add it to the client repository.
For more information, see [Extracting the public part of a self-signed certificate from a key repository on AIX, Linux, and Windows](#), and [Adding a CA certificate \(or the public part of a self-signed certificate\) into a key repository, on AIX, Linux, and Windows systems](#).
 - f) Extract the personal certificate from the client key repository and add it to the server key repository.
 - g) Alter the MQIPT.CONN.CHANNEL server connection channel to use TLS by using the MQSC command:

```
ALTER CHANNEL(MQIPT.CONN.CHANNEL) CHLTYPE(SVRCONN) TRPTYPE(TCP)
SSLCIPH(TLS_RSA_WITH_AES_128_CBC_SHA256)
```

2. On the MQIPT computer (see the diagram), copy the sample Java security manager policy to the MQIPT home directory, by entering the following command at a command prompt:

```
copy C:\mqipt\samples\mqiptSample.policy C:\mqiptHome\mqipt.policy
```

3. Start the Policy Tool utility by using the following command:

```
C:\mqipt\java\jre\bin\policytool
```

In the policy tool:

a) Click **File > Open** then select `C:\mqiptHome\mqipt.policy..`

b) Select:

```
file:/C:/Program Files/IBM/IBM MQ Internet Pass-Thru/lib/com.ibm.mq.ipt.jar
```

then click **Edit Policy Entry**

c) Change CodeBase from:

```
file:/C:/Program Files/IBM/IBM MQ Internet Pass-Thru/lib/com.ibm.mq.ipt.jar
```

to:

```
file:/C:/mqipt/lib/com.ibm.mq.ipt.jar
```

d) Change the file permissions for the IBM MQ Internet Pass-Thru, errors and logs directories from:

```
C:\Program Files\IBM\IBM MQ Internet Pass-Thru
```

to:

```
C:\mqiptHome
```

e) Change the other file permissions from:

```
C:\Program Files\IBM\IBM MQ Internet Pass-Thru
```

to:

```
C:\mqipt
```

f) Click **Add Permission**

Complete the fields as follows:

Permission: `java.net.SocketPermission`

Target: `client1.company1.com:1024-`

Actions: `accept, listen, resolve`

g) Click **File > Save** to save the changes to the policy file.

4. Edit `mqipt.conf`. Add the following properties to the `[global]` section and add the following route definition:

```
[global]
SecurityManager=true
SecurityManagerPolicy=C:\mqiptHome\mqipt.policy

[route]
ListenerPort=1415
Destination=server1.company2.com
DestinationPort=1414
SSLProxyMode=true
```

5. Start MQIPT.

Open a command prompt, and enter the following command:

```
C:\mqipt\bin\mqipt C:\mqiptHome -n ipt1
```

where `C:\mqiptHome` indicates the location of the MQIPT configuration file, `mqipt.conf`, and `ipt1` is the name to be given to the instance of MQIPT.

The following messages indicate that MQIPT has started successfully:

```
5724-H72 (C) Copyright IBM Corp. 2000, 2024. All Rights Reserved
MQCPI001 IBM MQ Internet Pass-Thru V9.4.0.0 starting
MQCPI004 Reading configuration information from mqipt.conf
MQCPI152 MQIPT name is ipt1
```

```

MQCPI055 Setting the java.security.policy to C:\mqiptHome\mqipt.policy
MQCPI053 Starting the Java Security Manager
MQCPI021 Password checking has been enabled on the command port
MQCPI011 The path C:\mqiptHome\mqipt\logs will be used to store the log files
MQCPI006 Route 1415 has started and will forward messages to :
MQCPI034 ....server1.company2.com(1414)
MQCPI035 ....using SSLProxyMode protocol
MQCPI078 Route 1415 ready for connection requests

```

- At a command prompt on the IBM MQ client system, enter the following command to run the TLS sample program:

```

AMQSSSLC -m MQIPT.QM1 -c MQIPT.CONN.CHANNEL -x 10.9.1.2(1415)
          -k "C:\ProgramData\IBM\MQ\clientkey" -l cert_label -s
          TLS_RSA_WITH_AES_128_CBC_SHA256

```

where `cert_label` is the label of the client certificate that you created in step “1.d” on page 193.

Using a security exit

In this scenario, you can use a supplied sample security exit, called `SampleSecurityExit`, so that only client connections that use a channel name starting with the characters `MQIPT.` are allowed.

Before you begin

- Before you start to use this scenario, make sure that you have completed the prerequisite tasks listed in “Getting started with IBM MQ Internet Pass-Thru” on page 162.
- Install Java 8.0 JDK.
- Add the Java `bin` subdirectory to the **PATH** environment variable.

About this task

The sample exit used in this scenario is `SampleSecurityExit.java`. It is provided with `MQIPT` in the `samples/exits` subdirectory of the `MQIPT` installation directory.

If you use the suggested server connection channel name of `MQIPT.CONN.CHANNEL` (as used in most of these scenarios), the client connection will be allowed to complete and an IBM MQ message can be placed on the queue.

To demonstrate that the security exit is working as expected, define another server connection channel with any name that does not start with the characters `MQIPT.` (for example, `TEST.CONN.CHANNEL`) and try the **amqspu**tc command again, but having changed the **MQSERVER** environment variable to use the new channel name. This time the connection will be refused and a 2059 (MQRC_Q_MGR_NOT_AVAILABLE) error will be returned.

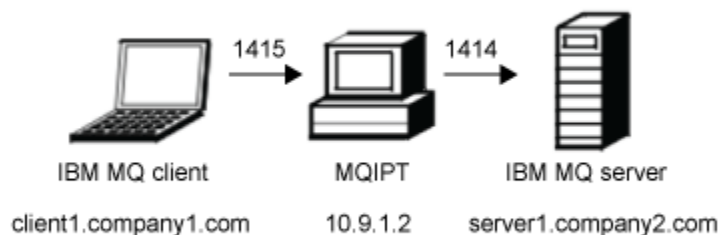


Figure 43. Security exit network diagram

This diagram shows the connection flow from the IBM MQ client (called `client1.company1.com` on port 1415) through `MQIPT` to the IBM MQ server (called `server1.company2.com` on port 1414).

Procedure

To use a security exit, complete the following steps:

1. On the MQIPT computer:

- a) Create a directory called `exits` in the MQIPT home directory by issuing the following command in a command prompt:

```
md C:\mqiptHome\exits
```

- b) Enter the following commands to compile the exit. You do not have to do this if you have not changed the exit code as the compiled sample exit is supplied with MQIPT.

```
C:
cd \mqipt\samples\exits
javac -classpath C:\mqipt\lib\com.ibm.mq.ipt.jar;. SampleSecurityExit.java
```

- c) Enter the following command to copy the compiled exit class file `SampleSecurityExit.class` to the `C:\mqiptHome\exits` directory:

```
copy C:\mqipt\samples\exits\SampleSecurityExit.class C:\mqiptHome\exits
```

- d) Edit `mqipt.conf` and add a route definition:

```
[route]
ListenerPort=1415
Destination=server1.company2.com
DestinationPort=1414
SecurityExit=true
SecurityExitName=SampleSecurityExit
```

- e) Open a command prompt and start MQIPT:

```
C:\mqipt\bin\mqipt C:\mqiptHome -n ipt1
```

where `C:\mqiptHome` indicates the location of the MQIPT configuration file, `mqipt.conf`, and `ipt1` is the name to be given to the instance of MQIPT.

The following messages indicate that MQIPT has started successfully:

```
5724-H72 (C) Copyright IBM Corp. 2000, 2024. All Rights Reserved
MQCPI001 IBM MQ Internet Pass-Thru V9.4.0.0 starting
MQCPI004 Reading configuration information from mqipt.conf
MQCPI152 MQIPT name is ipt1
MQCPI021 Password checking has been enabled on the command port
MQCPI011 The path C:\mqiptHome\logs will be used to store the log files
MQCPI006 Route 1415 has started and will forward messages to :
MQCPI034 ....server1.company2.com(1414)
MQCPI035 ....using MQ protocol
MQCPI079 ....using security exit C:\mqiptHome\exits\SampleSecurityExit
MQCPI080 .....and timeout of 30 seconds
MQCPI078 Route 1415 ready for connection requests
```

2. At a command prompt on the IBM MQ client system, enter the following commands:

- a) Set the **MQSERVER** environment variable:

```
SET MQSERVER=MQIPT.CONN.CHANNEL/tcp/10.9.1.2(1415)
```

- b) Put a message:

```
amqsputc MQIPT.LOCAL.QUEUE MQIPT.QM1
Hello world
```

Press Enter twice after typing the message string.

- c) Get the message:

```
amqsgetc MQIPT.LOCAL.QUEUE MQIPT.QM1
```

The message, "Hello world" is returned.

Routing client connection requests to IBM MQ queue manager servers by using security exits

In this scenario, you can dynamically route client connection requests, in a round-robin fashion, to a group of three IBM MQ queue manager servers. The queue manager on each server in the group must be identical.

Before you begin

- Before you start to use this scenario, make sure that you have completed the prerequisite tasks listed in [“Getting started with IBM MQ Internet Pass-Thru” on page 162.](#)
- Install Java 8.0 JDK.
- Add the Java bin subdirectory to the **PATH** environment variable.

About this task

The sample exit used in this scenario is `SampleRoutingExit.java`. It is provided with MQIPT in the `samples/exits` subdirectory of the MQIPT installation directory.

The name and location of the compiled exit class file is defined with the MQIPT **SecurityExitName** and **SecurityExitPath** properties.

The list of queue manager and server names to be used is read from a configuration file, called `SampleRoutingExit.conf`. The exit expects the configuration file to exist in the same directory as the exit class file.

The first time the **amqsputc** command is run, the IBM MQ message is placed on the MQIPT.LOCAL.QUEUE queue on the first server. The second time it is run, the message is placed on the queue on the second server, and so on. Using this setup, it is not possible for the **amqsgetc** command to retrieve the message just placed on the queue, because the client connection request used by the **amqsgetc** command is passed to the next queue in the list. However, running the **amqsputc** command three times, followed by three **amqsgetc** commands, ensures that each message is retrieved in the same order.

Of course, by using another IBM MQ client, connecting directly to a queue manager (that is, not using the MQIPT in this sample), you can selectively retrieve messages from any of the queue managers.

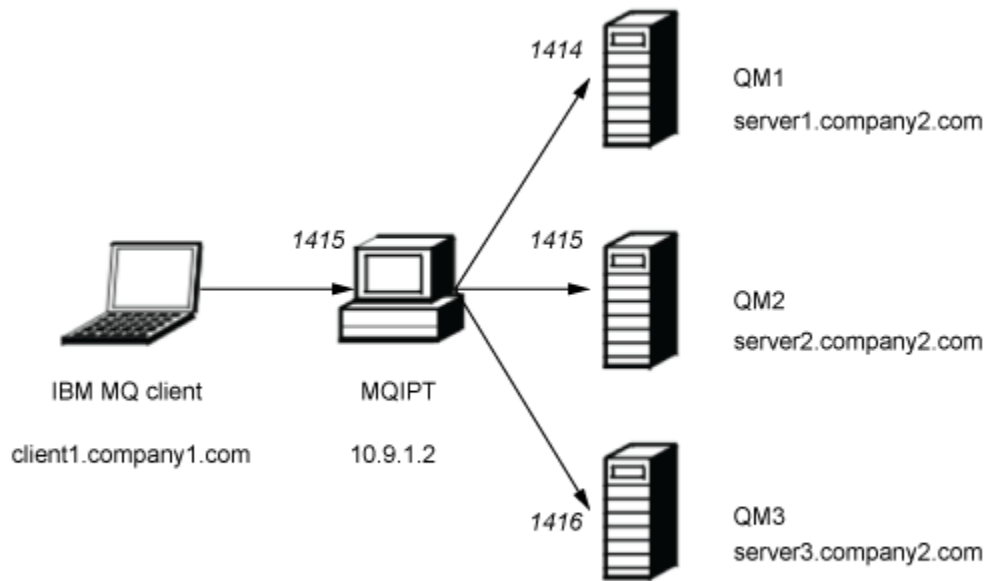


Figure 44. Routing security exit network diagram

This diagram shows the connection flow from the IBM MQ client (called client1.company1.com on port 1415) through MQIPT to three IBM MQ servers (called server1.company2.com, server2.company2.com, and server3.company2.com).

Procedure

To route client connection requests sequentially to three different IBM MQ queue manager servers by using security exits, complete the following steps:

1. Create three identical queue managers named MQIPT.QM1 on three separate servers.
Each queue manager has a SVRCONN channel named MQIPT.CONN.CHANNEL and an empty local queue named MQIPT.LOCAL.QUEUE.
2. On the MQIPT server:
 - a) Create a directory called exits in the MQIPT home directory by issuing the following command in a command prompt:

```
md C:\mqiptHome\exits
```

- b) In the C:\mqiptHome\exits directory (where C:\mqiptHome is the directory where the mqipt.conf file is located), create a sample configuration file, called SampleRoutingExit.conf that contains the names of your three queue managers.

For example, the configuration file could contain the following entries:

```
server1.company2.com:1414
server2.company2.com:1415
server3.company2.com:1416
```

Ensure that there are no blank lines before the first entry in the file and that each entry is a valid server name. If you have used different server names, change these names to match your environment.

- c) Open a command prompt and enter the following commands to compile the exit. You do not have to do this if you have not changed the exit code as the compiled sample exit is supplied with MQIPT.

```
C:
cd \mqipt\samples\exits
javac -classpath C:\mqipt\lib\com.ibm.mq.ipt.jar;. SampleRoutingExit.java
```

- d) Enter the following command to copy the compiled exit class file `SampleRoutingExit.class` to the `C:\mqiptHome\exits` directory:

```
copy C:\mqipt\samples\exits\SampleRoutingExit.class C:\mqiptHome\exits
```

- e) Edit `mqipt.conf` and add a route definition:

```
[route]
ListenerPort=1415
Destination=server1.company2.com
DestinationPort=1414
SecurityExit=true
SecurityExitPath=C:\mqiptHome\exits
SecurityExitName=SampleRoutingExit
```

Note that you do not have to set **SecurityExitPath** if you put `SampleRoutingExit.conf` in the default `C:\mqiptHome\exits` directory.

- f) Start MQIPT.

Open a command prompt and enter the following command:

```
C:\mqipt\bin\mqipt C:\mqiptHome -n ipt1
```

where `C:\mqiptHome` indicates the location of the MQIPT configuration file, `mqipt.conf`, and `ipt1` is the name to be given to the instance of MQIPT.

The following messages indicate that MQIPT has started successfully:

```
5724-H72 (C) Copyright IBM Corp. 2000, 2024. All Rights Reserved
MQCPI001 IBM MQ Internet Pass-Thru V9.4.0.0 starting
MQCPI004 Reading configuration information from mqipt.conf
MQCPI152 MQIPT name is ipt1
MQCPI021 Password checking has been enabled on the command port
MQCPI011 The path C:\mqiptHome\logs will be used to store the log files
MQCPI006 Route 1415 has started and will forward messages to :
MQCPI034 ...server1.company2.com(1414)
MQCPI035 ....using MQ protocol
MQCPI079 ....using security exit C:\mqiptHome\exits\SampleRoutingExit
MQCPI080 .....and timeout of 30 seconds
MQCPI078 Route 1415 ready for connection requests
```

3. At a command prompt on the IBM MQ client system, enter the following commands:

- a) Set the **MQSERVER** environment variable:

```
SET MQSERVER=MQIPT.CONN.CHANNEL/TCP/10.9.1.2(1415)
```

- b) Put three messages:

```
amqsputc MQIPT.LOCAL.QUEUE MQIPT.QM1
Hello world 1
amqsputc MQIPT.LOCAL.QUEUE MQIPT.QM1
Hello world 2
amqsputc MQIPT.LOCAL.QUEUE MQIPT.QM1
Hello world 3
```

Press Enter twice after typing each message string.

- c) Get the messages:

```
amqsgetc MQIPT.LOCAL.QUEUE MQIPT.QM1
amqsgetc MQIPT.LOCAL.QUEUE MQIPT.QM1
amqsgetc MQIPT.LOCAL.QUEUE MQIPT.QM1
```

The messages, `Hello world 1`, `Hello world 2`, and `Hello world 3` are returned.

Dynamically routing client connection requests

In this scenario, you can dynamically route client connection requests to a target server, based on the name of the channel being used.

Before you begin

- Before you start to use this scenario, make sure that you have completed the prerequisite tasks listed in [“Getting started with IBM MQ Internet Pass-Thru” on page 162.](#)
- Install Java 8.0 JDK.
- Add the Java bin subdirectory to the **PATH** environment variable.

About this task

If you use the name of the queue manager as the first part of the channel name, you only need to use one MQIPT route to service all connection requests. For example, to connect to QM1, the name of a SVRCONN channel might be QM1.MQIPT.CHANNEL.

The sample exit used in this scenario is `SampleOneRouteExit.java`. It is provided with MQIPT in the `samples/exits` subdirectory of the MQIPT installation directory.

The name and location of the compiled exit class file is defined with the MQIPT **SecurityExitName** and **SecurityExitPath** properties.

The list of queue manager and server names to be used is read from a configuration file, called `SampleOneRouteExit.conf`. The exit expects the configuration file to exist in the same directory as the exit class file.

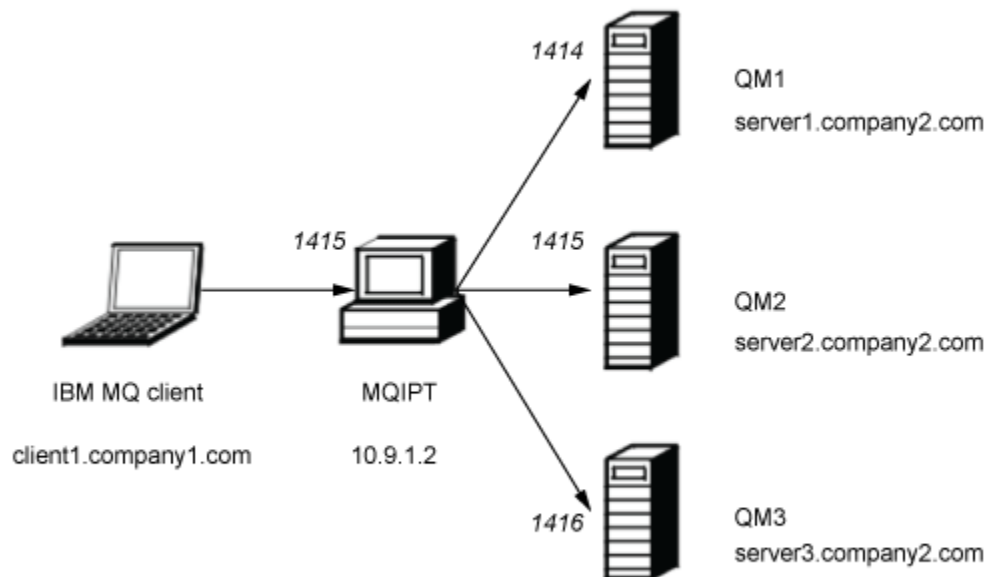


Figure 45. Dynamic one route exit network diagram

This diagram shows the connection flow from the IBM MQ client (called `client1.company1.com` on port 1415) through MQIPT to three IBM MQ servers (called `server1.company2.com`, `server2.company2.com`, and `server3.company2.com`).

Procedure

To dynamically route client connection requests, complete the following steps:

1. Create three different queue managers on three separate servers.

Each queue manager has a SVRCONN channel named after itself, for example, QM1.MQIPT.CHANNEL on queue manager QM1, and an empty local queue named MQIPT.LOCAL.QUEUE.

2. On the MQIPT server:

- a) Create a directory called exits in the MQIPT home directory by issuing the following command in a command prompt:

```
md C:\mqiptHome\exits
```

- b) In the C:\mqiptHome\exits directory (where C:\mqiptHome is the directory where the mqipt.conf file is located), create a sample configuration file, called SampleOneRouteExit.conf that contains the names of your three queue managers.

For example, the configuration file could contain the following entries:

```
server1.company2.com:1414
server2.company2.com:1415
server3.company2.com:1416
```

Ensure that there are no blank lines before the first entry in the file and that each entry is a valid server name. If you have used different server names, change these names to match your environment.

Note that all queue manager names in the list must be unique. If you list the same name more than once, even if the queue managers are on different servers, only the last entry for that name is registered.

- c) Open a command prompt and enter the following commands to compile the exit. You do not have to do this if you have not changed the exit code as the compiled sample exit is supplied with MQIPT.

```
C:
cd \mqipt\samples\exits
javac -classpath C:\mqipt\lib\com.ibm.mq.ipt.jar;. SampleOneRouteExit.java
```

- d) Enter the following command to copy the compiled exit class file SampleOneRouteExit.class to the C:\mqiptHome\exits directory:

```
copy C:\mqipt\samples\exits\SampleOneRouteExit.class C:\mqiptHome\exits
```

- e) Edit mqipt.conf and add the following route definition:

```
[route]
ListenerPort=1415
Destination=server1.company2.com
DestinationPort=1414
SecurityExit=true
SecurityExitName=SampleOneRouteExit
```

- f) Open a command prompt and start MQIPT:

```
C:\mqipt\bin\mqipt C:\mqiptHome -n ipt1
```

where C:\mqiptHome indicates the location of the MQIPT configuration file, mqipt.conf, and ipt2 is the name to be given to the instance of MQIPT.

The following messages indicate that MQIPT has started successfully:

```
5724-H72 (C) Copyright IBM Corp. 2000, 2024. All Rights Reserved
MQCPI001 IBM MQ Internet Pass-Thru V9.4.0.0 starting
MQCPI004 Reading configuration information from mqipt.conf
MQCPI152 MQIPT name is ipt2
MQCPI021 Password checking has been enabled on the command port
MQCPI011 The path C:\mqiptHome\logs will be used to store the log files
MQCPI006 Route 1415 has started and will forward messages to :
```

```
MQCPI034 ....server1.company2.com(1414)
MQCPI035 ....using MQ protocol
MQCPI079 ....using security exit C:\mqiptHome\exits\SampleOneRouteExit
MQCPI080 .....and timeout of 5 seconds
MQCPI078 Route 1415 ready for connection requests
```

3. At a command prompt on the IBM MQ client system, enter the following commands:

a) Set the **MQSERVER** environment variable:

```
SET MQSERVER=QM1.MQIPT.CHANNEL/TCP/10.9.1.2(1415)
```

b) Put a message:

```
amqsputc MQIPT.LOCAL.QUEUE QM1
Hello world 1
```

Press Enter twice after typing the message string.

The message is routed by MQIPT to QM1 as the SVRCONN channel name starts with QM1.

c) Get the message from QM1:

```
amqsgetc MQIPT.LOCAL.QUEUE QM1
```

The message, Hello world 1 is returned.

d) Reset the **MQSERVER** environment variable:

```
SET MQSERVER=QM2.MQIPT.CHANNEL/TCP/10.9.1.2(1415)
```

e) Put a message:

```
amqsputc MQIPT.LOCAL.QUEUE QM2
Hello world 2
```

Press Enter twice after typing the message string.

The message is routed by MQIPT to QM2 as the SVRCONN channel name starts with QM2.

f) Get the message from QM2:

```
amqsgetc MQIPT.LOCAL.QUEUE QM2
```

The message, Hello world 2 is returned.

g) Reset the **MQSERVER** environment variable again:

```
SET MQSERVER=QM3.MQIPT.CHANNEL/TCP/10.9.1.2(1415)
```

h) Put a message:

```
amqsputc MQIPT.LOCAL.QUEUE QM3
Hello world 3
```

Press Enter twice after typing the message string.

The message is routed by MQIPT to QM3 as the SVRCONN channel name starts with QM3.

i) Get the message from QM3:

```
amqsgetc MQIPT.LOCAL.QUEUE QM3
```

The message, Hello world 3 is returned.

Using a certificate exit to authenticate a TLS server

In this scenario, you can authenticate a TLS connection by using a certificate exit.

Before you begin

- Before you start to use this scenario, make sure that you have completed the prerequisite tasks listed in [“Getting started with IBM MQ Internet Pass-Thru” on page 162](#).
- Install Java 8.0 JDK.
- Add the Java bin subdirectory to the **PATH** environment variable.

About this task

This scenario performs the same function as the [“Authenticating a TLS server” on page 168](#) scenario, with the addition of a certificate exit.

The sample exit used in this scenario is `SampleCertificateExit.java`. It is provided with MQIPT in the `samples/exits` subdirectory of the MQIPT installation directory.

By changing the value of the **SSLExitData** property, the TLS connection between the two MQIPT servers can be allowed or rejected.

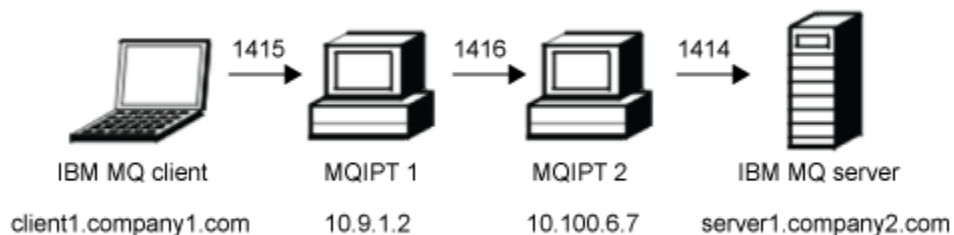


Figure 46. SSL/TLS server network diagram

This diagram shows the connection from the IBM MQ client (called `client1.company1.com` on port 1415) through two instances of MQIPT to the IBM MQ server (called `server1.company2.com` on port 1414).

Procedure

To use a certificate exit to authenticate an TLS server, complete the following steps:

1. On the MQIPT 1 system:

- a) Create a directory called `exits` in the MQIPT home directory by issuing the following command in a command prompt:

```
md C:\mqiptHome\exits
```

- b) Open a command prompt and enter the following commands to compile the exit. You do not have to do this if you have not changed the exit code as the compiled sample exit is supplied with MQIPT.

```
C:
cd \mqipt\samples\exits
javac -classpath C:\mqipt\lib\com.ibm.mq.ipt.jar;. SampleCertificateExit.java
```

- c) Enter the following command to copy the compiled exit class file `SampleCertificateExit.class` to the `C:\mqiptHome\exits` directory:

```
copy C:\mqipt\samples\exits\SampleCertificateExit.class C:\mqiptHome\exits
```

d) Edit mqipt.conf and add the following route definition:

```
[route]
ListenerPort=1415
Destination=9.100.6.7
DestinationPort=1416
SSLClient=true
SSLClientKeyRing=C:\mqipt\samples\ssl\sslSample.pfx
SSLClientKeyRingPW=<mqiptPW>1!PCaB1HwrFMOp43ngjwgArg==!6N/vsbqru7iqMhFN+wozxQ==
SSLClientExit=true
SSExitName=SampleCertificateExit
SSExitPath=C:\mqiptHome\exits
SSExitData=allow
```

e) Open a command prompt and start MQIPT:

```
C:\mqipt\bin\mqipt C:\mqiptHome -n ipt1
```

where C:\mqiptHome indicates the location of the MQIPT configuration file, mqipt.conf, and ipt1 is the name to be given to the instance of MQIPT.

The following messages indicate that MQIPT has started successfully:

```
5724-H72 (C) Copyright IBM Corp. 2000, 2024. All Rights Reserved
MQCPI001 IBM MQ Internet Pass-Thru V9.4.0.0 starting
MQCPI004 Reading configuration information from mqipt.conf
MQCPI152 MQIPT name is ipt1
MQCPI021 Password checking has been enabled on the command port
MQCPI011 The path C:\mqiptHome\logs will be used to store the log files
MQCPI006 Route 1415 has started and will forward messages to :
MQCPI034 ....9.100.6.7(1416)
MQCPI035 ....using MQ protocol
MQCPI036 ....SSL Client side enabled with properties :
MQCPI031 .....CipherSuites <null>
MQCPI032 .....keyring file C:\mqipt\samples\ssl\sslSample.pfx
MQCPI047 .....CA keyring file <null>
MQCPI038 .....peer certificate uses UID=*,CN=*,T=*,OU=*,DC=*,O=*,
                                     STREET=*,L=*,ST=*,PC=*,C=*,DNQ=*
MQCPI129 .....using certificate exit C:\mqiptHome\exits\SampleCertificateExit
MQCPI131 .....and certificate exit data 'allow'
MQCPI078 Route 1415 ready for connection requests
```

2. On the MQIPT 2 system:

a) Edit mqipt.conf and add the following route definition:

```
[route]
ListenerPort=1416
Destination=Server1.company2.com
DestinationPort=1414
SSLServer=true
SSLServerKeyRing=C:\mqipt\samples\ssl\sslSample.pfx
SSLServerKeyRingPW=C:\mqipt\samples\ssl\sslSample.pwd
```

b) Open a command prompt and start MQIPT:

```
C:
cd \mqipt\bin
mqipt .. -n ipt2
```

where .. indicates that the MQIPT configuration file, mqipt.conf, is in the parent directory, and ipt2 is the name to be given to the instance of MQIPT.

The following messages indicate that MQIPT has started successfully:

```
5724-H72 (C) Copyright IBM Corp. 2000, 2024. All Rights Reserved
MQCPI001 IBM MQ Internet Pass-Thru V9.4.0.0 starting
MQCPI004 Reading configuration information from mqipt.conf
MQCPI152 MQIPT name is ipt2
MQCPI021 Password checking has been enabled on the command port
MQCPI011 The path C:\mqipt\logs will be used to store the log files
MQCPI006 Route 1416 has started and will forward messages to :
MQCPI034 ....server1.company2.com(1414)
MQCPI035 ....using MQ protocol
MQCPI037 ....SSL Server side enabled with properties :
MQCPI031 .....CipherSuites <null>
```

```
MQCPI032 .....key ring file C:\mqipt\samples\ssl\sslSample.pfx
MQCPI047 .....CA key ring file <null>
MQCPI038 .....peer certificate uses UID=*,CN=*,T=*,OU=*,DC=*,O=*,
                                STREET=*,L=*,ST=*,PC=*,C=*,DNQ=*
MQCPI033 .....client authentication set to false
MQCPI078 Route 1416 ready for connection requests
```

3. At a command prompt on the IBM MQ client system, enter the following commands:

a) Set the **MQSERVER** environment variable:

```
SET MQSERVER=MQIPT.CONN.CHANNEL/tcp/10.9.1.2(1415)
```

b) Put a message:

```
amqsputc MQIPT.LOCAL.QUEUE MQIPT.QM1
Hello world
```

Press Enter twice after typing the message string.

c) Get the message:

```
amqsgetc MQIPT.LOCAL.QUEUE MQIPT.QM1
```

The message, "Hello world" is returned.

MQ Adv.

V 9.4.0

MQ Adv. VUE

Kafka Connect scenarios

With IBM MQ and Apache Kafka specializing in different aspects of the messaging spectrum, one on connectivity and the other on data, solutions often require data to flow between the two. You can achieve this using Kafka Connect.

Kafka Connect provides a framework for moving data from an external system into a Kafka cluster, or from a Kafka cluster into an external system. This is achieved by connectors.

There are many different types of connectors available, and IBM provides connectors for use with IBM MQ. Connectors come in two different types:

- Source connectors transfer data from an external system into Kafka.


The IBM MQ source connector consumes messages from an IBM MQ queue and publishes them as events into a Kafka topic.

- Sink connectors transfer data into an external system from Kafka.

The IBM MQ sink connector consumes events from a Kafka topic and sends them as messages to an MQ queue.

See [Kafka Connect and connectors](#) for more information.

Kafka Connect scenarios might include:

- A core banking system with IBM MQ used as the connectivity backbone. You want to take a copy of messages moving through IBM MQ and push them into Kafka for analytics
- You want to extend your core banking system to emit data into Kafka, but only want data to enter Kafka when the banking transaction completes successfully, so use IBM MQ as a transactional bridge
-  You need to get data into z/OS from Multiplatforms. Multiplatform development team has experience with Kafka, z/OS team want to exploit IBM MQ integration with CICS / IMS

From IBM MQ 9.4.0, if your enterprise has IBM MQ Advanced entitlement, IBM MQ Advanced for z/OS VUE entitlement, IBM MQ Advanced for Multiplatforms entitlement, or IBM MQ Appliance entitlement, you get access to IBM provided, and supported source and sink connectors.

Notes:

1. These approaches can be used with any variant of Kafka, for example, Apache Kafka and IBM Event Streams.

2. Support is provided only for the two IBM connectors, not the Kafka Connect framework itself.

MQ Adv.

V 9.4.0

MQ Adv. VUE

Kafka Connect common topologies

This section describes the three approaches that can be used when integrating IBM MQ with Kafka through the IBM connectors.

See [“Obtaining the Connectors” on page 212](#) for more information on obtaining the connectors, and [“Using the Connectors” on page 212](#) for more information on queue manager connection and configuration options.

Direct to queue (source)

Applications which want to send data to Kafka using IBM MQ can send those messages to the queue used by the IBM MQ source connector. The IBM MQ source connector then takes those messages and transfers them to the relevant Kafka topic.

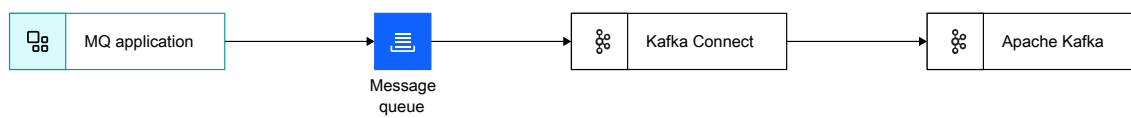


Figure 47. Direct to queue (source)

This approach should be used when an application needs to send data into Kafka and that data is not already being sent to IBM MQ.

Sending data using IBM MQ means that the send of the message can be done inside an transaction coordinated with other updates, for example to a database. This approach also avoids the need to set up a potentially short lived connection to Kafka, and instead using an existing connection to IBM MQ.

Streaming queue copy (source)

In many cases there is a need to take a copy of existing data moving through IBM MQ and sending it into Kafka, for example, for analytics. From IBM MQ 9.3 this can be achieved using streaming queues. Streaming queues allow messages being put to one queue to be copied by the queue manager to a second queue, without affecting the applications using the first queue. See [“Streaming queues” on page 29](#) for more information.

For example:

```
DEF QL(TO.APP) STREAMQ(TO.KAFKA) STRMQOS(MUSTDUP)
DEF QL(TO.KAFKA)
```

means that when a message is sent to TO . APP, a copy of that message must be sent to TO . KAFKA. The IBM MQ source connector then takes those messages from TO.KAFKA and transfers them to the relevant Kafka topic.

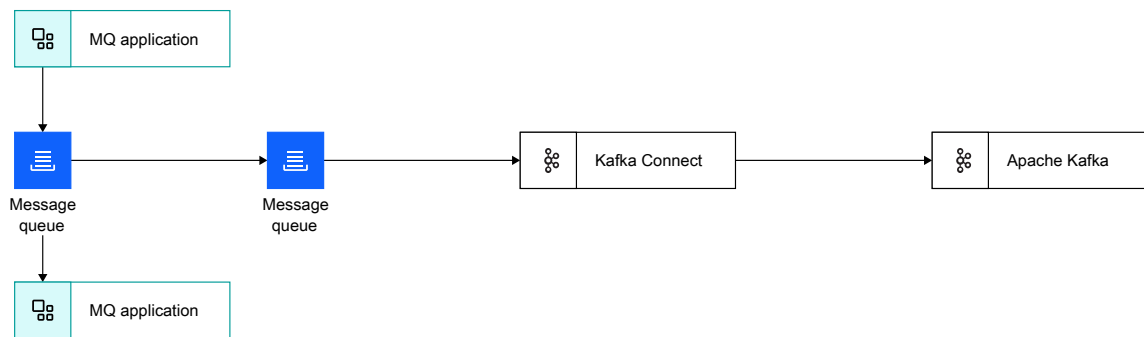


Figure 48. Streaming queue copy (source)

Enabling streaming queues has no effect on existing applications, as the original message does not change. The message sent to the second queue is identical to the original message with the same payload, message ID, correlation ID, and so on.

Direct to queue (sink)

As with the source connector, the sink connector can be configured to receive data from a Kafka topic into a queue directly.

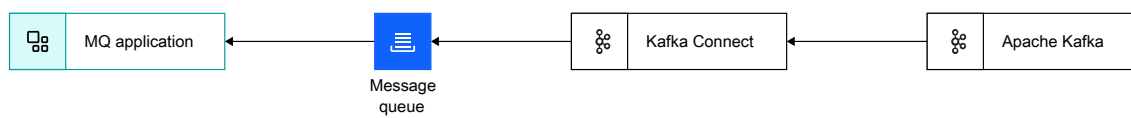


Figure 49. Direct to queue (sink)

Receiving data through IBM MQ means that the receive of the message can be done inside an transaction coordinated with other updates, for example to a database.

This approach also avoids the need to set up a potentially short lived connection to Kafka, and instead using an existing connection to IBM MQ.

Obtaining the Connectors

V 9.4.0 The version 2 connectors that are shipped with IBM MQ 9.4.0 provide at-least-once, and exactly-once, message delivery.

V 9.4.0 For more information on the differences between at-least-once and exactly-once delivery, and how to configure exactly-once delivery, see [“Exactly once support” on page 214](#).

z/OS In IBM MQ Advanced for z/OS Value Unit Edition and IBM MQ Advanced for z/OS, the connectors and their samples are provided in the kafka-connect directory of the Connector Pack component, in z/OS UNIX System Services (USS).

In IBM MQ Advanced for Multiplatforms and IBM MQ Appliance, these connectors and required configuration files can be retrieved by logging in to Fix Central and searching for V . R . M . F - IBM - MQ - Kafka - Connectors . tar . gz, for example, 9 . 4 . 0 . 0 - IBM - MQ - Kafka - Connectors . tar . gz.

MQ Adv. **MQ Adv. VUE** **MQ Adv. z/OS** These are the connectors shipped with each IBM MQ version:

IBM MQ version number	IBM MQ for Multiplatforms tar file name	Source connector version	Sink connector version	Exactly-once delivery support
V 9.4.0 V 9.4.0 9.4.0	9.4.0.0-IBM-MQ-Kafka_Connectors.tar.gz	2.0.0	2.1.0	Yes

Notes:

1. You should always take the most recent version of the connectors and check regularly for updates. The connectors provided with IBM MQ are the latest at the time when the product ships, and are periodically updated to the most recent version.
2. If support for the IBM MQ Connectors is provided through entitlement to IBM MQ Advanced for z/OS Value Unit Edition, IBM MQ Advanced for Multiplatforms or IBM MQ Appliance, then the connectors must be connected to a queue manager running with that entitlement.

Using the Connectors

The connectors are configured using either properties or JSON files. Sample files are provided with the connectors.

Details on the configuration options along with how to set up the connectors are provided at:

Source Connector: [Kafka Connect source connector for IBM MQ](#)

Sink Connector: [Kafka Connect sink connector for IBM MQ](#)

V 9.4.0 To enable exactly-once support in the source connector see [Running the MQ source connector](#), and for the sink connector see [Running the MQ sink connector](#).

V 9.4.0 For more information on the differences between at-least-once and exactly-once delivery, and how to configure exactly-once delivery, see [“Exactly once support” on page 214](#).

For Kafka Connect to run the IBM MQ Connectors, it needs to have the connector jars, and various IBM MQ jar files, on its class path. The following jar files are required:

```
jms.jar
com.ibm.mq.allclient.jar
org.json.jar
> V 9.4.0 bcpkix-jdk18on.jar
> V 9.4.0 bcprov-jdk18on.jar
> V 9.4.0 bcutil-jdk18on.jar
```

For example:

Source connector

```
> V 9.4.0 > V 9.4.0 :
```

```
export CLASSPATH=$CLASSPATH:/path-to-kafka-jars/kafka-connect-mq-source-2.0.0.jar:
/path-to-mq-jars/jms.jar:/path-to-mq-jars/com.ibm.mq.allclient.jar:/path-to-mq-jars/org.json.jar:
/path-to-mq-jars/bcpkix-jdk18on.jar:/path-to-mq-jars/bcprov-jdk18on.jar:/path-to-mq-jars/bcutil-
jdk18on.jar
```

Sink connector

```
> V 9.4.0 > V 9.4.0 :
```

```
export CLASSPATH=$CLASSPATH:/path-to-kafka-jars/kafka-connect-mq-sink-2.1.0.jar:
/path-to-mq-jars/jms.jar:/path-to-mq-jars/com.ibm.mq.allclient.jar:/path-to-mq-jars/org.json.jar:
/path-to-mq-jars/bcpkix-jdk18on.jar:/path-to-mq-jars/bcprov-jdk18on.jar:/path-to-mq-jars/bcutil-
jdk18on.jar
```

where:

path-to-kafka-jars is the path to the location where the IBM MQ connectors are installed
 path-to-mq-jars is the path to the location where the IBM JMS client is installed.

> V 9.4.0 > V 9.4.0 The 2.1.0 version of the sink connector includes the ability to add values to the MQMD before sending messages to IBM MQ. See [Enabling MQMD in IBM MQ sink connector v2](#) for more information.

> z/OS If running on z/OS, USS_ROOT/kafka-connect/source/kafka-connect-mq-source.jar, in the Connector Pack component, points to the most recent version of the source connector, and USS_ROOT/kafka-connect/sink/kafka-connect-mq-sink.jar points to the most recent version of the sink connector.

Kafka Connect, and the IBM MQ connectors can be run on any platform with a Java Virtual machine. They do not have to run on the same platform as the queue managers, or the Kafka cluster that they connect to.

However, if there is a long distance between the queue managers and the Kafka clusters, you should position the connectors relatively close to the queue managers; ideally in the same availability zone, or data center.

Using the Connectors on z/OS

> z/OS

The connectors are fully supported with queue managers running on all platforms, including on z/OS. Connections to z/OS queue managers can be either through a server-connection channel, or through local bindings.

In performance testing environments on IBM z/OS and IBM MQ for z/OS, optimal performance has been obtained by running the connectors on z/OS in z/OS UNIX System Services (USS) and connecting to queue managers using local bindings. Details on these findings are available here: [Kafka Connectors for IBM MQ – an MQ for z/OS perspective](#).

Running Kafka Connect in USS on z/OS requires some extra setup steps; documentation for these is here: [Running connectors on IBM z/OS](#).

There are two versions of the IBM MQ Kafka Connectors, 1 and 2. The version 2 connectors provide support for exactly-once and at-least-once message delivery, whereas the version 1 connectors provide support for at-least-once message delivery.

At-least-once message delivery means that in the event of a failure, in either IBM MQ, the IBM MQ Kafka Connector, or Kafka:

- For the source connector, IBM MQ messages are not lost, but might be delivered to Kafka multiple times resulting in duplicate Kafka messages.
- For the sink connector, Kafka messages are not lost, but might be delivered to IBM MQ multiple times resulting in duplicate IBM MQ messages.

Exactly-once message delivery means that in the event of a failure, in either IBM MQ, the IBM MQ Kafka Connector, or Kafka:

- For the source connector, IBM MQ messages are not lost, and will be delivered to Kafka with no chance for duplicate Kafka messages.
- For the sink connector, Kafka messages are not lost, and will be delivered to IBM MQ with no chance for duplicate IBM MQ messages.

Exactly-once support is only available in the version 2 connector shipped with IBM MQ, or with IBM Event Streams. It is not available in the version 1 connector.

The version 2 connector can run in either at-least-once or exactly-once mode. Exactly-once support is enabled by appropriate configuration of Kafka, and by making use of a "*state queue*". Each instance of a connector running in exactly-once mode needs its own state queue.

Throughput and scalability of the connectors running in exactly-once mode are less than running in at-least-once mode. Only enable exactly-once mode if your applications aren't designed to work with duplicate messages.

See [Running the MQ source connector](#) for details on configuring exactly-once mode in the source connector, and [Running the MQ sink connector](#) for details on configuring exactly-once mode in the sink connector.

Notices

This information was developed for products and services offered in the U.S.A.

IBM may not offer the products, services, or features discussed in this document in other countries. Consult your local IBM representative for information on the products and services currently available in your area. Any reference to an IBM product, program, or service is not intended to state or imply that only that IBM product, program, or service may be used. Any functionally equivalent product, program, or service that does not infringe any IBM intellectual property right may be used instead. However, it is the user's responsibility to evaluate and verify the operation of any non-IBM product, program, or service.

IBM may have patents or pending patent applications covering subject matter described in this document. The furnishing of this document does not grant you any license to these patents. You can send license inquiries, in writing, to:

IBM Director of Licensing
IBM Corporation
North Castle Drive
Armonk, NY 10504-1785
U.S.A.

For license inquiries regarding double-byte (DBCS) information, contact the IBM Intellectual Property Department in your country or send inquiries, in writing, to:

Intellectual Property Licensing
Legal and Intellectual Property Law
IBM Japan, Ltd.
19-21, Nihonbashi-Hakozakicho, Chuo-ku
Tokyo 103-8510, Japan

The following paragraph does not apply to the United Kingdom or any other country where such provisions are inconsistent with local law: INTERNATIONAL BUSINESS MACHINES CORPORATION PROVIDES THIS PUBLICATION "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. Some states do not allow disclaimer of express or implied warranties in certain transactions, therefore, this statement may not apply to you.

This information could include technical inaccuracies or typographical errors. Changes are periodically made to the information herein; these changes will be incorporated in new editions of the publication. IBM may make improvements and/or changes in the product(s) and/or the program(s) described in this publication at any time without notice.

Any references in this information to non-IBM websites are provided for convenience only and do not in any manner serve as an endorsement of those websites. The materials at those websites are not part of the materials for this IBM product and use of those websites is at your own risk.

IBM may use or distribute any of the information you supply in any way it believes appropriate without incurring any obligation to you.

Licensees of this program who wish to have information about it for the purpose of enabling: (i) the exchange of information between independently created programs and other programs (including this one) and (ii) the mutual use of the information which has been exchanged, should contact:

IBM Corporation
Software Interoperability Coordinator, Department 49XA
3605 Highway 52 N
Rochester, MN 55901
U.S.A.

Such information may be available, subject to appropriate terms and conditions, including in some cases, payment of a fee.

The licensed program described in this information and all licensed material available for it are provided by IBM under terms of the IBM Customer Agreement, IBM International Program License Agreement, or any equivalent agreement between us.

Any performance data contained herein was determined in a controlled environment. Therefore, the results obtained in other operating environments may vary significantly. Some measurements may have been made on development-level systems and there is no guarantee that these measurements will be the same on generally available systems. Furthermore, some measurements may have been estimated through extrapolation. Actual results may vary. Users of this document should verify the applicable data for their specific environment.

Information concerning non-IBM products was obtained from the suppliers of those products, their published announcements or other publicly available sources. IBM has not tested those products and cannot confirm the accuracy of performance, compatibility or any other claims related to non-IBM products. Questions on the capabilities of non-IBM products should be addressed to the suppliers of those products.

All statements regarding IBM's future direction or intent are subject to change or withdrawal without notice, and represent goals and objectives only.

This information contains examples of data and reports used in daily business operations. To illustrate them as completely as possible, the examples include the names of individuals, companies, brands, and products. All of these names are fictitious and any similarity to the names and addresses used by an actual business enterprise is entirely coincidental.

COPYRIGHT LICENSE:

This information contains sample application programs in source language, which illustrate programming techniques on various operating platforms. You may copy, modify, and distribute these sample programs in any form without payment to IBM, for the purposes of developing, using, marketing or distributing application programs conforming to the application programming interface for the operating platform for which the sample programs are written. These examples have not been thoroughly tested under all conditions. IBM, therefore, cannot guarantee or imply reliability, serviceability, or function of these programs.

If you are viewing this information softcopy, the photographs and color illustrations may not appear.

Programming interface information

Programming interface information, if provided, is intended to help you create application software for use with this program.

This book contains information on intended programming interfaces that allow the customer to write programs to obtain the services of IBM MQ.

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

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

Trademarks

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

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

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

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

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

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



Part Number:

(1P) P/N: