IBM MQ

# Internet Pass-Thru

*Version 2 Release 1*

SC34-2920-00

IBM MQ

# Internet Pass-Thru

*Version 2 Release 1*

SC34-2920-00

**Note**

Before using this information and the product it supports, read the information in "Notices" on page 177.

**Edition Notice**

This edition applies to version 2.1.0.3 of IBM MQ Internet Pass-Thru and to all subsequent releases and modifications until otherwise indicated in new editions.

# Contents

**Figures . . . . . . . . . . . . . . . . . v**

**Chapter 1. Introduction to IBM MQ Internet Pass-Thru . . . . . . . . . . 1**
Edition Notice . . . . . . . . . . . . . . 4
  Who this book is for . . . . . . . . . . . 4
  What you need to know to understand this document . . . . . . . . . . . . . . . 4
  Prerequisites . . . . . . . . . . . . . . 4
  Accessibility information . . . . . . . . . 5
  Privacy . . . . . . . . . . . . . . . . 5

**Chapter 2. What's new in MQIPT Version 2.1 . . . . . . . . . . . . . . . . . 7**

**Chapter 3. How MQIPT works . . . . . 11**
Starting and stopping MQIPT . . . . . . . . 11
Possible configurations of MQIPT . . . . . . . 13
Compatible configurations . . . . . . . . . 16
Supported channel configurations . . . . . . . 17
Channel termination and failure conditions . . . . 18
Safety of messages . . . . . . . . . . . . 18
Multi-instance queue managers and high availability 18

**Chapter 4. Using MQIPT features . . . 21**
HTTP support . . . . . . . . . . . . . . 21
  HTTPS . . . . . . . . . . . . . . . . 22
SOCKS support . . . . . . . . . . . . . 22
  Clustering . . . . . . . . . . . . . . 23
SSL/TLS support . . . . . . . . . . . . . 24
  SSL/TLS handshake . . . . . . . . . . . 31
  MQIPT implementation of SSL/TLS . . . . . 32
  Encrypting a keyring password . . . . . . . 32
  Selecting certificates from a key-ring file . . . . 33
  Trust settings . . . . . . . . . . . . . 33
  Testing SSL/TLS. . . . . . . . . . . . . 34
  SSL/TLS error messages . . . . . . . . . 35
  iKeyman . . . . . . . . . . . . . . . 37
  Certificate exit . . . . . . . . . . . . 40
  LDAP and CRLs. . . . . . . . . . . . . 45
  Multi-valued certificate Distinguished Name OU properties . . . . . . . . . . . . . . . 46
Network Dispatcher support . . . . . . . . 47
Java Security Manager . . . . . . . . . . . 49
Security exits . . . . . . . . . . . . . . 52
  The com.ibm.mq.ipt.exit.SecurityExit class . . . 53
  The com.ibm.mq.ipt.exit.SecurityExitResponse class . . . . . . . . . . . . . . . . 55
  Security exit return codes . . . . . . . . . 56
  Tracing . . . . . . . . . . . . . . . . 56
Port number control . . . . . . . . . . . 57
Other security considerations . . . . . . . . 57
Connection logs . . . . . . . . . . . . . 58

**Chapter 5. Installing, uninstalling, and migrating MQIPT . . . . . . . . . . . 59**
Installing MQIPT . . . . . . . . . . . . . 59
Uninstalling MQIPT version 2.0 . . . . . . . 61
Uninstalling MQIPT version 2.1 . . . . . . . 62
Migrating from MQIPT Version 2.0 to Version 2.1. . 62
Upgrading your MQIPT Version 2.1 installation . . 66

**Chapter 6. Scenarios: Getting started with MQIPT . . . . . . . . . . . . . 69**
Scenario: Verifying that MQIPT is working correctly 70
Scenario: Creating a key-ring file . . . . . . . 72
Scenario: Creating test certificates . . . . . . . 75
Scenario: Authenticating an SSL/TLS server . . . 76
Scenario: Authenticating an SSL/TLS client. . . . 79
Scenario: Configuring HTTP tunneling . . . . . 81
Scenario: Configuring an HTTP proxy . . . . . 83
Scenario: Configuring access control . . . . . . 85
Scenario: Configuring a SOCKS proxy . . . . . 87
Scenario: Configuring a SOCKS client . . . . . 88
Scenario: Configuring MQIPT clustering support . . 90
Scenario: Allocating port numbers . . . . . . . 93
Scenario: retrieving CRLs by using an LDAP server 94
Scenario: running MQIPT in SSL/TLS proxy mode 97
Scenario: running MQIPT in SSL/TLS proxy mode with a security manager . . . . . . . . . . 99
Scenario: Apache rewrite . . . . . . . . . 101
Scenario: Using a security exit. . . . . . . . 103
Scenario: Routing client connection requests to IBM MQ queue manager servers by using security exits. 105
Scenario: Dynamically routing client connection requests . . . . . . . . . . . . . . . 108
Scenario: Using a certificate exit to authenticate an SSL/TLS server. . . . . . . . . . . . . 111

**Chapter 7. Administering and configuring MQIPT . . . . . . . . . 115**
Using the MQIPT Administration Client . . . . 115
  Starting the Administration Client . . . . . 115
  Administering an instance of MQIPT . . . . 116
  Setting MQIPT properties . . . . . . . . 116
  Administration Client menu options. . . . . 117
Administering MQIPT by using the command line 118
Configuration reference information . . . . . . 119
  Summary of properties . . . . . . . . . 120
  Global properties . . . . . . . . . . . 124
  Route properties . . . . . . . . . . . 125
Making backups . . . . . . . . . . . . . 142
Performance tuning . . . . . . . . . . . 143

**Chapter 8. Troubleshooting and support . . . . . . . . . . . . . . 145**
Automatically starting MQIPT. . . . . . . . 145
  Using a Windows service control program. . . 147

Using a UNIX or Linux init.d system service 148
Checking for end-to-end connectivity . . . . . 148
Using JRE diagnostic options . . . . . . . . . 149
Tracing errors in MQIPT. . . . . . . . . . . 149
Tracing errors in iKeyman and iKeycmd . . . . 150
Reporting problems . . . . . . . . . . . . 150

**Chapter 9. Messages . . . . . . . . . 153**

List of MQIPT MQC messages . . . . . . . . 153

**Index . . . . . . . . . . . . . . . . 173**

**Notices . . . . . . . . . . . . . . . 177**

Programming interface information . . . . . . . 179
Trademarks . . . . . . . . . . . . . . . . 179

**Sending your comments to IBM . . . 181**

# Figures

1. Example of MQIPT as a channel concentrator 2
2. Example of MQIPT in a DMZ firewall . . . . 3
3. Example of MQIPT and HTTP tunneling . . . 3
4. Example of MQIPT and SSL/TLS . . . . . 4
5. Using the Network Dispatcher with MQIPT 48
6. Installation verification test network diagram 71
7. SSL/TLS server network diagram . . . . . 77
8. SSL/TLS client network diagram . . . . . 79
9. HTTP proxy network diagram . . . . . . 81
10. HTTP proxy network diagram . . . . . . 83
11. Access control network diagram . . . . . 85
12. SOCKS proxy network diagram . . . . . . 87
13. SOCKS client network diagram . . . . . . 89
14. Clustering network diagram . . . . . . . 90
15. Port allocation network diagram . . . . . 93
16. LDAP server network diagram . . . . . . 95
17. SSL/TLS proxy mode network diagram 98
18. SSL/TLS proxy mode network diagram 99
19. Apache rewrite network diagram . . . . . 102
20. Security exit network diagram . . . . . . 104
21. Routing security exit network diagram 106
22. Dynamic one route exit network diagram 109
23. SSL/TLS server network diagram . . . . . 111

# Chapter 1. Introduction to IBM MQ Internet Pass-Thru

IBM® MQ Internet Pass-Thru (MQIPT) is an extension to the base IBM IBM MQ product. MQIPT (category 3 SupportPac MS81) can be downloaded from the IBM MQ SupportPac website (http://www.ibm.com/software/integration/wmq/supportpacs).

MQIPT runs as a stand-alone service that can receive and forward IBM MQ message flows, either between two IBM MQ queue managers or between a IBM MQ client and a IBM MQ queue manager.

MQIPT enables this connection when the client and server are not on the same physical network.

One or more instances of MQIPT can be placed in the communication path between two IBM MQ queue managers, or between a IBM MQ client and a IBM MQ queue manager. The instances of MQIPT allow the two IBM MQ systems to exchange messages without needing a direct TCP/IP connection between the two systems. This is useful if the firewall configuration prohibits a direct TCP/IP connection between the two systems.

MQIPT listens on one or more TCP/IP ports for incoming connections, which can carry either normal IBM MQ messages, IBM MQ messages tunneled inside HTTP, or messages encrypted using Transport Layer Security (TLS) or Secure Sockets Layer (SSL). MQIPT can handle multiple concurrent connections.

The IBM MQ channel that makes the initial TCP/IP connection request is referred to as the *caller*, the channel to which it is attempting to connect as the *responder*, and the queue manager that it is ultimately trying to contact as the *destination queue manager*.

MQIPT holds data in memory as it forwards it from its source to its destination. No data is saved on disk (except for memory paged to disk by the operating system). The only time MQIPT accesses the disk explicitly is to read its configuration file and to write connection log and trace records.

The full range of IBM MQ channel types can be made through one or more instances of MQIPT. The presence of MQIPT in a communication path has no effect on the functional characteristics of the connected IBM MQ components, but there might be some effect on the performance of message transfer.

MQIPT can be used in conjunction with IBM MQ and IBM Integration Bus, as described in "Possible configurations of MQIPT" on page 13.

There are a number of potential uses for MQIPT:

## MQIPT can be used as a channel concentrator

By using MQIPT in this way, channels to or from multiple separate hosts can appear to a firewall as if they are all to or from the MQIPT host. This makes it easier to define and manage firewall filtering rules.

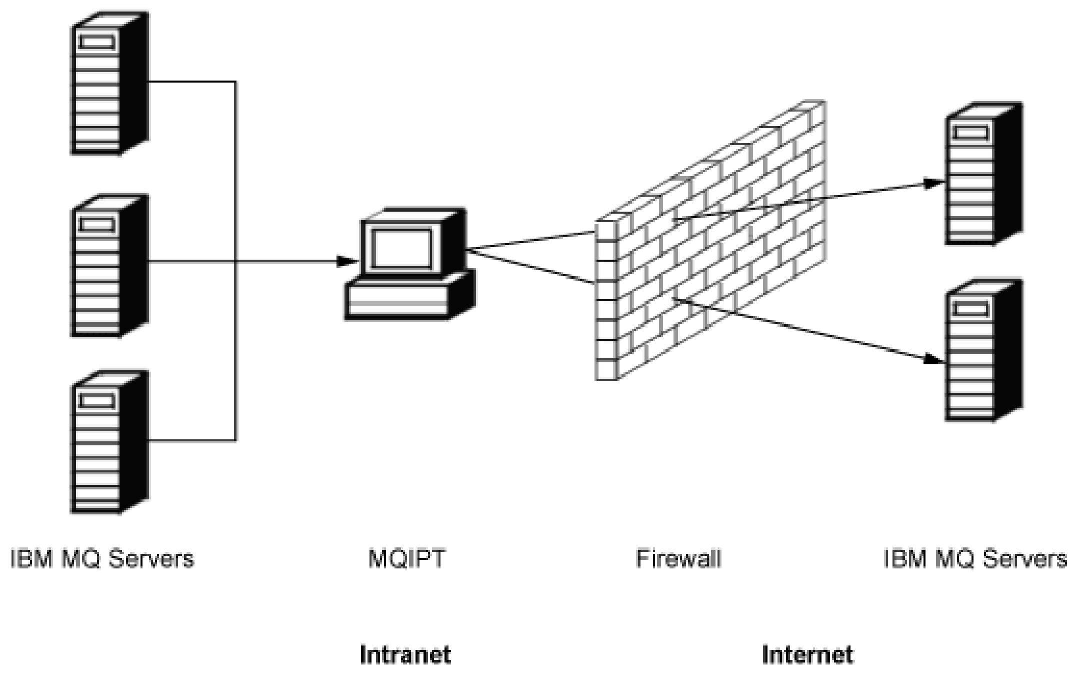


*Figure 1. Example of MQIPT as a channel concentrator*

### MQIPT can be placed in a DMZ to provide a single point of access

If MQIPT is placed within a DMZ firewall (a firewall configuration for securing local area networks), on a computer with a known and trusted internet protocol (IP) address, MQIPT can be used to listen for incoming IBM MQ channel connections which it can then forward to the trusted intranet; the inner firewall must allow this trusted computer to make inbound connections. In this configuration, MQIPT prevents external requests for access from receiving the true IP addresses of the computers in the trusted intranet. In this way, MQIPT provides a single point of access.

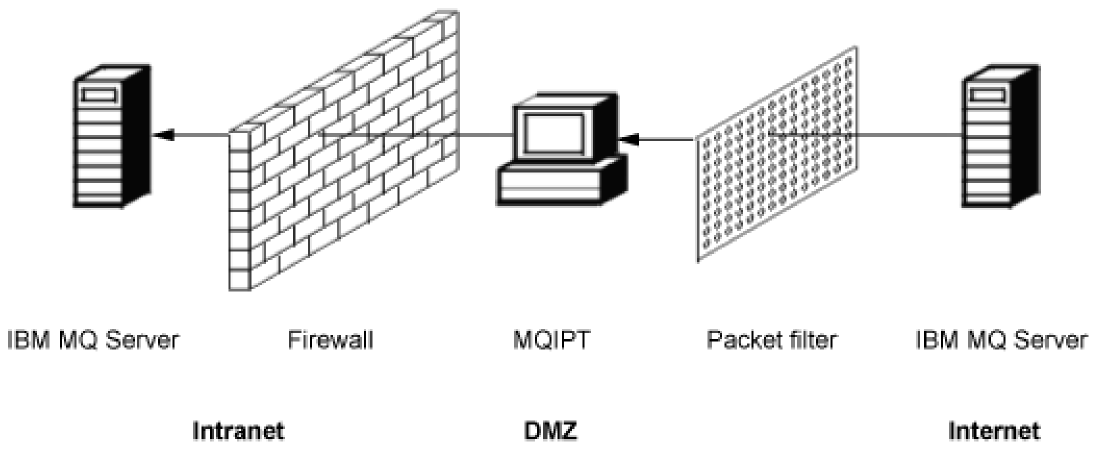


*Figure 2. Example of MQIPT in a DMZ firewall*

## MQIPT can communicate by means of HTTP tunneling

If two instances of MQIPT are deployed in line, they can communicate by using HTTP. The HTTP tunneling feature enables requests to be transmitted through firewalls, by the use of existing HTTP proxies. The first MQIPT inserts the IBM MQ protocol into HTTP and the second extracts the IBM MQ protocol from its HTTP wrapper and forwards it to the destination queue manager.

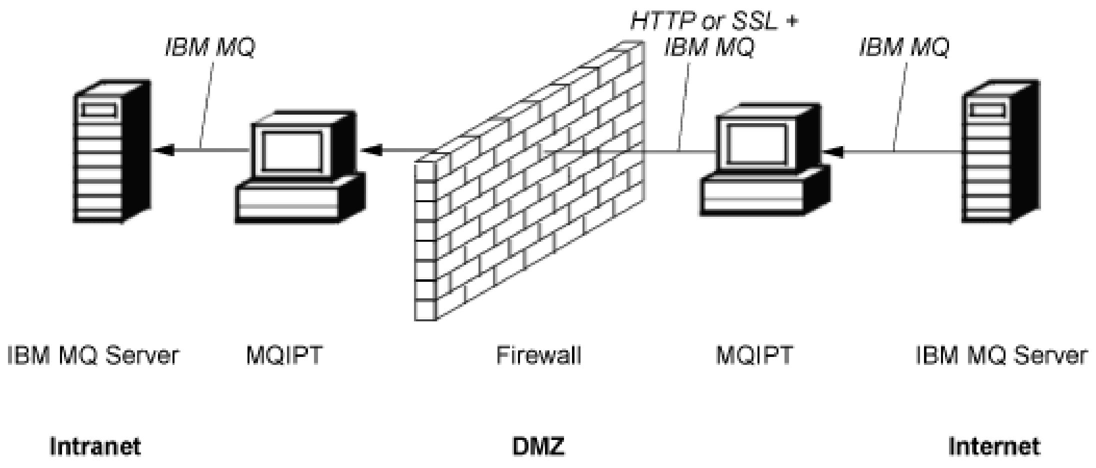


*Figure 3. Example of MQIPT and HTTP tunneling*

## MQIPT can encrypt messages

If MQIPT is configured as in the previous example, requests can be encrypted before transmission through firewalls. The first MQIPT encrypts the data and the second decrypts it using SSL/TLS before sending it to the destination queue manager.

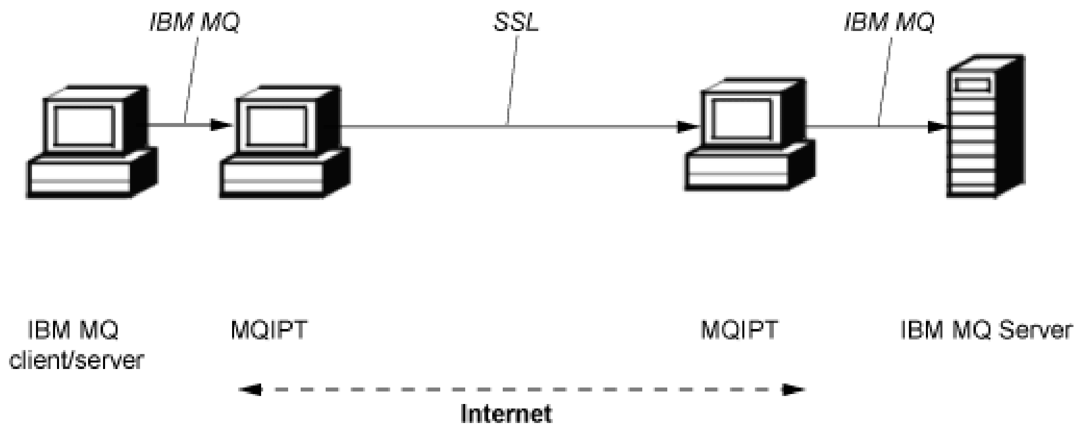


*Figure 4. Example of MQIPT and SSL/TLS*

## Edition Notice

This edition applies to version 2.1.0.3 of IBM MQ Internet Pass-Thru 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.

### Who this book is for

This book is for systems designers, technical IBM MQ administrators, and firewall and network administrators.

### What you need to know to understand this document

You need a good understanding of the following concepts:

- The administration of IBM MQ queue managers and message channels, as described in the IBM MQ product documentation.
- The way that firewalls are implemented.
- Internet protocol routing/networking.
- The IBM Network Dispatcher for load balancing and enhanced availability.

### Prerequisites

The readme file supplied with MQIPT specifies the exact operating system versions that are supported.

The J2SE Development Kit (JDK) 6.0 or later is required to create an MQIPT exit.

The only supported network protocol is TCP/IP.

## Accessibility information

The Administration Client graphical user interface (GUI) has been built with accessibility in mind. It is straightforward to perform all of the available functions without using a mouse, by using keyboard equivalents. You can navigate round the screen by using Tab, Shift+Tab, Ctrl+Tab, and the Up Arrow, Down Arrow, Left Arrow, and Right Arrow keys in the standard manner. You can achieve the equivalent to clicking buttons first selecting the button and then pressing Enter.

You can reach menu options either by combinations of Tab and Arrow keys or by using the accelerator keys, which are available for all the options. For example, you can close the GUI by selecting first Alt+f, then Alt+q (File->Quit). When you reach a menu item, you can activate it by using Enter.

You can use the Arrow keys to navigate around the tree. In particular, you can use the Right and Left arrow keys to open or close an MQIPT node, allowing you to show or hide the routes.

You can use the Spacebar key to change the state of selected checkboxes. You can use the Enter key to select fields for editing.

### Look and feel

Ideally the GUI should adopt the look and feel of the environment. As this is not always possible, you can provide a configuration file to tailor the look and feel of the GUI to suit your needs. The configuration file is called `custom.properties` and should be placed in the `bin` subdirectory.

Use this configuration file to configure the following GUI elements:

- The foreground color (the color of the text)
- The background color
- The font of the text
- The style of the text (plain, bold, italic, or bold and italic)

A sample configuration file `customSample.properties` is provided, which contains comments showing how it can be changed. You are encouraged to copy this file to `bin/custom.properties` and to make any required changes.

## Privacy

IBM Software products, including software as a service solutions, ("Software Offerings") might use cookies or other technologies to collect product usage information, to help improve the user experience, to tailor interactions with the user, or for other purposes. In many cases no personally identifiable information is collected by the Software Offerings. Some of our Software Offerings can help you to collect personally identifiable information. The information that follows is specific to this Software Offering.

The IBM MQ protocol requires full duplex data flows over HTTP. To satisfy this requirement, the HTTP features of IBM MQ Internet Pass-Thru use cookies to coordinate pairs of related network connections. These cookies record the following information:

- An anonymous session identifier
- The MQIPT route destination IP address or DNS host name
- The MQIPT route destination TCP/IP port number

Use of HTTP is disabled by default. If HTTP is enabled, cookies are required for correct MQIPT operation.

MQIPT can optionally record connection log files of network connections. These files might include the following information:

- IP addresses and DNS host names
- TCP/IP port numbers
- Digital certificate identity information (but not private keys)
- Additional information logged by security exits or certificate exits, if they are used. IBM supplies some sample exits with MQIPT which record IP addresses, DNS host names, TCP/IP port numbers and IBM MQ channel names. You can also use other exits which might record other details. By default, no exits are enabled.

The connection log is an optional feature of MQIPT. It is disabled by default, although it is enabled in the supplied sample configuration file `mqiptSample.conf`. Any configuration based upon the sample is therefore likely to use a connection log. The connection log can be disabled by setting the **`ConnectionLog`** property to `false` in the `[global]` section of `mqipt.conf`. Connection log files are never automatically transferred outside the system where MQIPT is running; they are only written to the local disk.

MQIPT can also optionally record trace files for problem diagnosis. These files might include the following information:

- IP addresses and DNS host names
- TCP/IP port numbers
- Digital certificate identity information (but not private keys)
- IBM MQ object names, such as queue manager names, channel names, and queue names
- The content of any IBM MQ messages that flow through MQIPT (except in **`SSLProxyMode`** routes when the connections are encrypted)

The trace facility is an optional feature of MQIPT that is disabled by default. Trace can also be disabled in `mqipt.conf` by removing all **`Trace`** lines or changing all **`Trace`** lines to `Trace=0` (where zero indicates that trace is disabled). Trace files are never automatically transferred outside of the system where MQIPT is running; they are only written to the local disk.

# Chapter 2. What's new in MQIPT Version 2.1

## Provision of a Java™ Runtime Environment

MQIPT Version 2.1 includes a Java Runtime Environment (JRE). This means that you no longer have to provide a JRE on the **PATH** in order to run MQIPT. The MQIPT command scripts automatically uses the JRE provided. Only a JRE supplied by IBM for use with MQIPT should be installed. Using an alternative JRE is not supported.

The SSL and TLS support for MQIPT is provided by using the JSSE library from the supplied JRE.

## SSL/TLS features

MQIPT now supports several new SSL/TLS features:
- TLS 1.1 and TLS 1.2 protocol support.
- SHA-2 hash algorithms for digital signatures and CipherSuite message integrity. SHA-224, SHA-256, SHA-384 and SHA-512 are all supported.
- Elliptic Curve encryption.
- Support for many new CipherSuites including those that use Galois/Counter Mode (GCM).

See "SSL/TLS support" on page 24 for more information.

## Certificate and key management

MQIPT version 2.1 provides the same iKeyman and iKeycmd tools used to administer digital certificates in IBM MQ. These can be run using the new **mqiptKeyman** and **mqiptKeycmd** commands. For more information about MQIPT digital certificate considerations, see "Digital certificate considerations for MQIPT" on page 38.

## Multiple installations

From version 2.1 of MQIPT, you can install the product wherever you want on your computer, and can have several installations at the same time. Each installation can be used and maintained separately, so for example you can have different fix pack levels of MQIPT installed in different locations if you choose. See "Installing MQIPT" on page 59 for more details.

## Certificate DN attributes

The following additional certificate DN attributes are now supported for use in selecting site certificates and matching remote peer certificates:
- Domain Component (DC)
- Domain Name Qualifier (DNQ)
- Postal Code (PC)
- Street address (STREET)
- Title (T)
- User ID (UID)

**Note:** The iKeyman and iKeycmd tools refer to the postal code attribute as `POSTALCODE`, not `PC`. In particular, always specify `POSTALCODE` in the **`-dn`** parameter when you use the **`mqiptKeycmd`** command-line certificate management tool to request or create certificates with a postal code.

## New route properties

The following new route properties can be used:
- SSLClientDN_DC
- SSLClientDN_DNQ
- SSLClientDN_PC
- SSLClientDN_Street
- SSLClientDN_T
- SSLClientDN_UID
- SSLClientSiteDN_DC
- SSLClientSiteDN_DNQ
- SSLClientSiteDN_PC
- SSLClientSiteDN_Street
- SSLClientSiteDN_T
- SSLClientSiteDN_UID
- SSLServerDN_DC
- SSLServerDN_DNQ
- SSLServerDN_PC
- SSLServerDN_Street
- SSLServerDN_T
- SSLServerDN_UID
- SSLServerSiteDN_DC
- SSLServerSiteDN_DNQ
- SSLServerSiteDN_PC
- SSLServerSiteDN_Street
- SSLServerSiteDN_T
- SSLServerSiteDN_UID

See "Route properties" on page 125 for a full list of route properties with descriptions.

The **`SSLClientDN_DC`**, **`SSLClientSiteDN_DC`**, **`SSLServerDN_DC`** and **`SSLServerSiteDN_DC`** route properties can match multiple domain component (DC) values in certificate Distinguished Names. To match multiple DC values, use a comma as a separator in the route property value.

## Tracing

In version 2.0, tracing was global to the entire MQIPT process. The trace level was calculated as the maximum value of the **`Trace`** property from all sections in the `mqipt.conf` file: trace was then enabled or disabled for all threads in the process based on this value. It was not possible to trace a subset of routes, resulting in potentially large trace files.

In version 2.1, the **`Trace`** setting is route-specific. Enabling trace for one route by adding a **`Trace`** property to its `[route]` section in `mqipt.conf` does not cause other routes to be traced.

Routes without a **`Trace`** property in their `[route]` section inherit the trace setting from the `[global]` section. Therefore you can use the `[global]` section Trace property to enable trace for multiple routes, although any route that explicitly sets `Trace=0` is not traced because the `[route]` section trace setting overrides the `[global]` section setting. For more information about the global and route **`Trace`** property settings, see "Global properties" on page 124 and "Route properties" on page 125.

## Error messages

There are some new and amended error messages in Version 2.1. For a complete list of messages see, "List of MQIPT MQC messages" on page 153.

The SSL and TLS error messages have changed due to the use of JSSE. See "SSL/TLS error messages" on page 35 for details.

## Removal of the MQIPT servlet

The MQIPT servlet has been removed. The servlet supplied with MQIPT Version 2.0 can be downloaded separately and can still be used if necessary. Note that the servlet does not support the sharing conversations feature of IBM MQ, so any SVRCONN channels that connect through MQIPT must have SHARECNV(0).

# Chapter 3. How MQIPT works

In its simplest configuration, MQIPT acts as a IBM MQ protocol forwarder. It listens on a TCP/IP port and accepts connection requests from IBM MQ channels.

If a well-formed request is received, MQIPT establishes a further TCP/IP connection between itself and the destination IBM MQ queue manager. It then passes all protocol packets it receives from its incoming connection on to the destination queue manager, and it returns protocol packets from the destination queue manager back to the original incoming connection.

No change to the IBM MQ protocol (client/server or queue manager to queue manager) is involved because neither end is directly aware of the presence of the intermediary. New versions of the IBM MQ client or server code are not required.

To use MQIPT, the caller channel must be configured to use the MQIPT host name and port, not the host name and port of the destination queue manager. This is defined with the **CONNAME** property of the IBM MQ channel. MQIPT reads the incoming data and simply passes it through to the destination queue manager. Other configuration fields, such as the user ID and password in a client/server channel, are similarly passed to the destination queue manager.

### Multiple queue managers

MQIPT can be used to allow access to more than one destination queue manager. For this to work, there must be a mechanism to tell MQIPT which queue manager to connect to, so MQIPT uses the incoming TCP/IP port number to determine which queue manager to connect to.

You can therefore configure MQIPT to listen on multiple TCP/IP ports. Each listening port is mapped to a destination queue manager through an MQIPT *route*. You can define up to 100 such routes, which associate a listening TCP/IP port with the host name and port of the destination queue manager. This means that the host name (IP address) of the destination queue manager is never visible to the originating channel. Each route can handle multiple connections between its listening port and destination, each connection acting independently.

### MQIPT configuration file

MQIPT uses a configuration file called `mqipt.conf`. This file contains definitions of all routes and their associated properties. See Chapter 7, "Administering and configuring MQIPT," on page 115 for more information about `mqipt.conf`.

When MQIPT is launched, it starts each route that is listed in the configuration file. Messages are written to the system console showing the status of each route. When message MQCPI078 is shown for a route, that route is ready to accept connection requests.

## Starting and stopping MQIPT

You can start MQIPT either from the command line, or make it start automatically when the system is started. You can stop MQIPT by using either the Administration Client or the command line.

## Starting MQIPT from the command line

MQIPT is installed into an installation directory, such as:

- `/opt/mqipt`, with executable scripts in `/opt/mqipt/bin`
- `C:\MQIPT`, with executable scripts in `C:\MQIPT\bin`

MQIPT also uses a home directory, which contains the configuration file `mqipt.conf` and any files that are output by MQIPT when it is running. The following subdirectories of the MQIPT home directory are created automatically when MQIPT is invoked for the first time:

- An `errors` directory in which any First Failure Support Technology™ (FFST™) and trace files are written
- A `logs` directory in which the connection log is kept

The user ID under which MQIPT runs must have permission to create these directories, or alternatively the directories must already exist and the user ID must have permission to create, read, and write files in them. Also, if you are using a Security Manager policy then the security policy must grant the required permissions for these directories. For more information about Security Manager policy settings refer to "Java Security Manager" on page 49.

You can use the installation directory as a home directory. If you use this directory, you must ensure that the user ID under which MQIPT runs has the appropriate permissions, and that any Security Manager policy is configured correctly.

To start MQIPT, enter the following command:

- On Windows systems:

  ```
  MQIPT_INSTALLATION_PATH\bin\mqipt MQIPT_HOME_DIR
  ```
- On UNIX and Linux systems:

  ```
  MQIPT_INSTALLATION_PATH/bin/mqipt MQIPT_HOME_DIR
  ```

Note that the MQIPT home directory can be specified as either an absolute path or relative to the current working directory of the command shell.

Console messages show the status of MQIPT. If an error occurs, see Chapter 8, "Troubleshooting and support," on page 145. The following messages are an example of the output when MQIPT starts successfully:

```
5639-L92 (C) Copyright IBM Corp. 2000, 2013 All Rights Reserved
MQCPI001 IBM WebSphere MQ Internet Pass-Thru V2.1.0.0 starting
MQCPI004 Reading configuration information from mqipt.conf
MQCPI021 Password checking has been enabled on the command port
MQCPI008 Listening for control commands on port 1881
MQCPI011 The path C:\MQIPT\logs will be used to store the log files
MQCPI006 Route 1414 is starting and will forward messages to :
MQCPI034 ....examplehost(1414)
MQCPI035 ....using MQ protocols
MQCPI057 ....trace level 5 enabled
MQCPI078 Route 1414 ready for connection requests
```

## Starting MQIPT automatically

To start MQIPT automatically when the system is started, use the mqiptService command to install the MQIPT service.

- On Windows systems, see "Using a Windows service control program" on page 147.

- On other platforms, see "Using a UNIX or Linux init.d system service" on page 148.

### Stopping MQIPT

You can stop MQIPT by using either the Administration Client or the command line.

- To use the Administration Client, mqiptGui, see "Using the MQIPT Administration Client" on page 115.
- To use the command line administration tool, mqiptAdmin, see "Administering MQIPT by using the command line" on page 118.

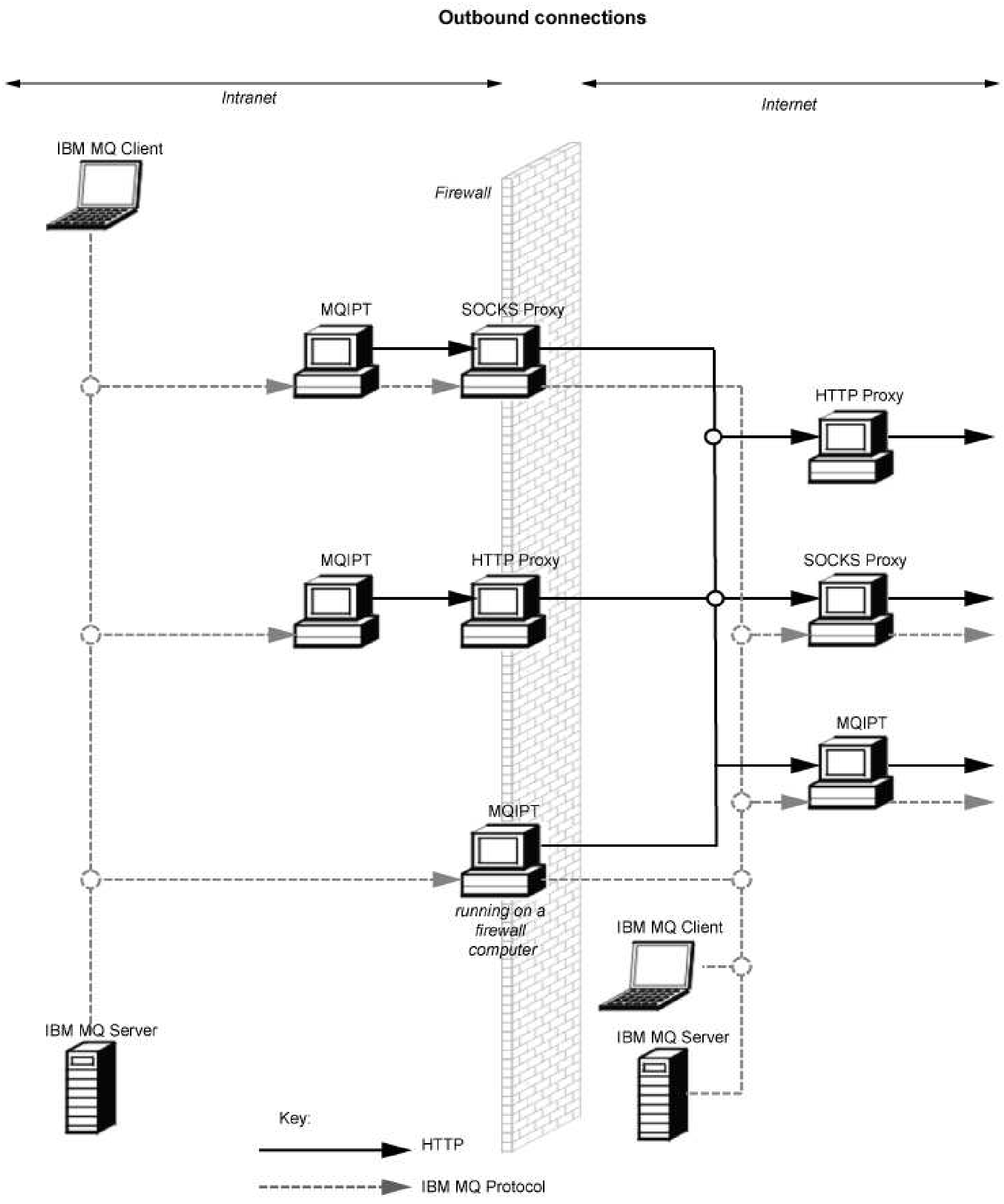
## Possible configurations of MQIPT

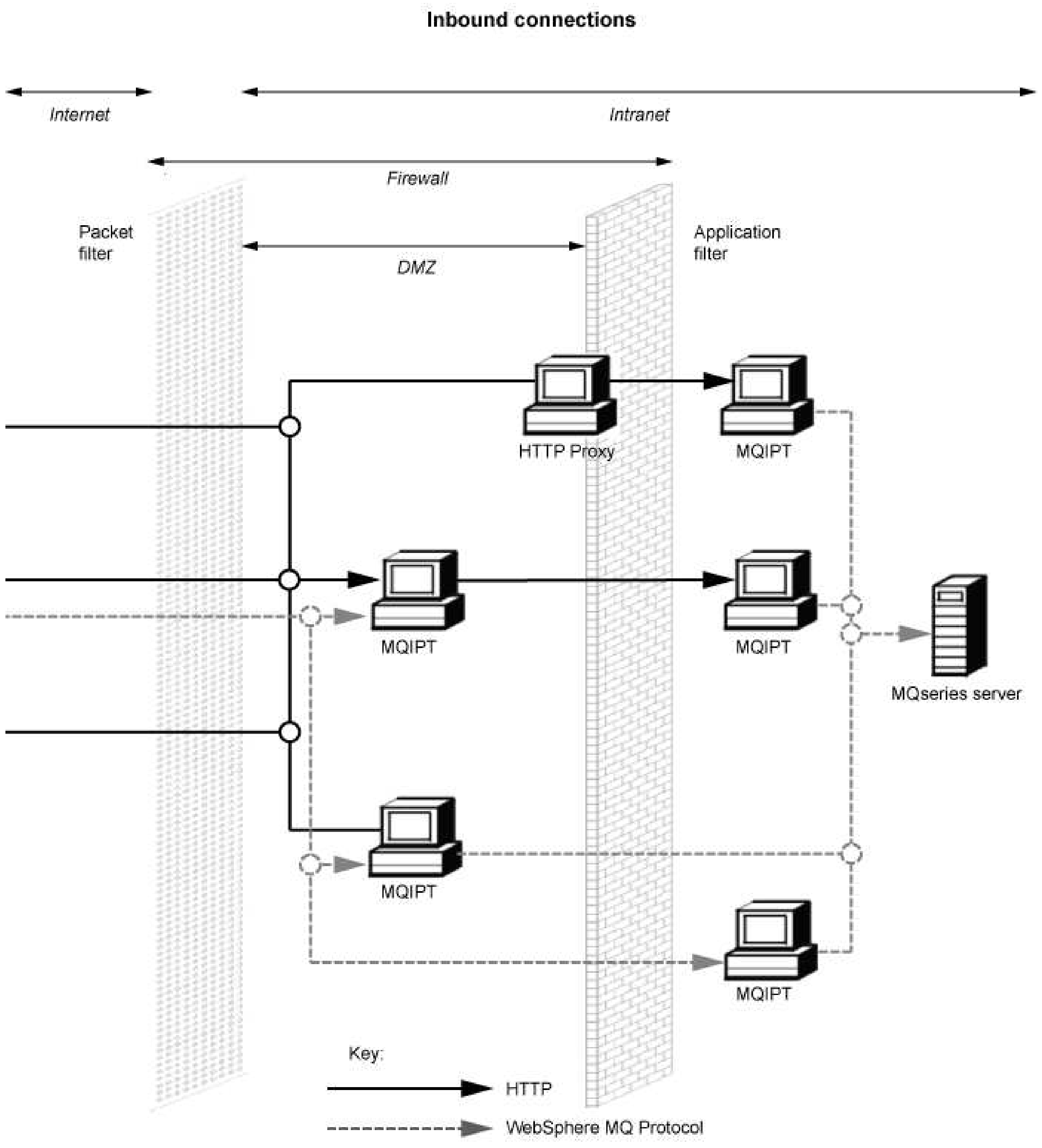
MQIPT can be used in conjunction with IBM MQ and IBM Integration Bus.

The following multi-part figure shows many of the possible configurations for MQIPT in a IBM MQ topology. It illustrates different ways in which MQIPT can send messages. It shows clients and servers on an intranet, inside a firewall, and on the Internet outside the firewall, passing messages to MQIPT, HTTP proxy, or SOCKS proxy, which forwards them.

The messages are received by an MQIPT proxy or an HTTP proxy in a DMZ before passing the message through the inbound firewall to a server.

Note that the HTTP proxy, SOCKS proxy, and MQIPT computers on the intranet side of the firewall represent the possibility of multiple computers chained together on the internet. For example, an MQIPT computer could communicate through one or more SOCKS or HTTP proxy computers, or further MQIPT computers, before reaching its target.


Outbound connections
Intranet
Internet
IBM MQ Client
Firewall
MQIPT
SOCKS Proxy
HTTP Proxy
MQIPT
HTTP Proxy
SOCKS Proxy
MQIPT
MQIPT
running on a firewall computer
IBM MQ Client
IBM MQ Server
IBM MQ Server
Key:
HTTP
IBM MQ Protocol


Inbound connections
Internet
Intranet
Firewall
Packet filter
Application filter
DMZ
HTTP Proxy
MQIPT
MQIPT
MQIPT
MQIPT
MQIPT
MQIPT
MQseries server
Key:
HTTP
WebSphere MQ Protocol

## Compatible configurations

Compatible connection scenarios where a IBM MQ client or queue manager communicates with MQIPT. The same or a second MQIPT route is used to communicate with a destination queue manager.

### Compatible configurations with a single MQIPT route

You can use a single MQIPT route to communicate with IBM MQ.

The columns in Table 1 contain the following information:
1. The protocol used between IBM MQ and the MQIPT route. The connection can be created by either a IBM MQ client or queue manager, and can use either IBM MQ Formats and Protocols (FAP) or a SSL/TLS protocol.
2. The mode in which the MQIPT route operates. The format of the communication across the Internet between MQIPT and IBM MQ, is determined by the configuration of the MQIPT route. Note that where the table mentions SSL, you can also use TLS.
3. The protocol used between the MQIPT route and the destination queue manager.

*Table 1. Valid configurations with a single instance of MQIPT*

| 1. IBM MQ source protocol | 2. Mode of the MQIPT route | 3. IBM MQ destination protocol |
|---|---|---|
| FAP | FAP-proxy (default) | FAP |
| | FAP-server and SSL-client | SSL/TLS |
| SSL/TLS | SSL-proxy | SSL/TLS |
| | SSL-server and FAP-client | FAP |
| | SSL-server and SSL-client | SSL/TLS |

### Compatible configurations with more than one MQIPT route

You might choose to use more than one route, on one or more instances of MQIPT, to communicate with IBM MQ.

The columns in Table 2 on page 17 contain the following information:
1. The protocol used between IBM MQ and the first MQIPT route. The connection can be created by either a IBM MQ client or queue manager, and can use either IBM MQ Formats and Protocols (FAP) or a SSL/TLS protocol.
2. The mode in which the first MQIPT route operates. The format of the communication across the Internet between MQIPT and IBM MQ, is determined by the configuration of the MQIPT route. Note that where the table mentions SSL, you can also use TLS.
3. The mode in which the second MQIPT route operates.
4. The protocol used between the second MQIPT route and the destination queue manager.

*Table 2. Valid configurations with multiple instances of MQIPT*

| 1. IBM MQ source protocol | 2. Mode of the first MQIPT route | 3. Mode of the second MQIPT route | 4. IBM MQ destination protocol |
|---|---|---|---|
| FAP (default) | FAP-proxy (default) | FAP-proxy (default) | FAP |
| | FAP-server and SSL-client | SSL-proxy | SSL/TLS |
| | | SSL-server and FAP-client | FAP |
| | | SSL-server and SSL-client | SSL/TLS |
| | HTTP-client | HTTP-server and SSL-client | SSL/TLS |
| | HTTPS-client | HTTPS-server and SSL-client | SSL/TLS |
| | HTTP-client | HTTP-server | FAP |
| | HTTPS-client | HTTPS-server | FAP |
| SSL/TLS | SSL-proxy | SSL-proxy | SSL/TLS |
| | | SSL-server and FAP-client | FAP |
| | | SSL-server and SSL-client | SSL/TLS |
| | HTTP-client | HTTP-server | FAP |
| | HTTPS-client | HTTPS-server | SSL/TLS |
| | HTTP-client | HTTP-server and SSL-client | FAP |
| | HTTPS-client | HTTPS-server and SSL-client | SSL/TLS |

## Supported channel configurations

All IBM MQ channel types are supported, but configuration is restricted to TCP/IP connections. To a IBM MQ client or queue manager, MQIPT appears as if it is the destination queue manager. Where channel configuration requires a destination host and port number, the MQIPT host name and listener port number are specified.

**Client/server channels**
MQIPT listens for incoming client connection requests, and then forwards them by using either HTTP tunneling, SSL/TLS, or as standard IBM MQ protocol packets. If MQIPT is using HTTP tunneling or SSL/TLS it forwards them on a connection to a second MQIPT. If it is not using HTTP tunneling, it forwards them on a connection to what it sees as the destination queue manager (although this could in turn be a further MQIPT). When the destination queue manager has accepted the client connection, packets are relayed between client and server.

**Cluster sender/receiver channels**
If MQIPT receives an incoming request from a cluster-sender channel, it assumes the queue manager has been SOCKS-enabled and the true destination address will be obtained during the SOCKS handshaking process. It forwards

the request to the next MQIPT or destination queue manager in exactly the same way as for client connection channels. This also includes auto-defined cluster-sender channels.

**Sender/receiver**
If MQIPT receives an incoming request from a sender channel, it forwards it to the next MQIPT or destination queue manager in exactly the same way as for client connection channels. The destination queue manager validates the incoming request and starts the receiver channel if appropriate. All communications between sender and receiver channel (including security flows) are relayed.

**Requester/server**
This combination is handled in the same manner as the preceding configurations. Validation of the connection request is performed by the server channel at the destination queue manager.

**Requester/sender**
The "callback" configuration could be of use if the two queue managers are not allowed to establish direct connections to each other, but are both allowed to connect to MQIPT and to accept connections from it.

**Server/requester and server/receiver**
These are handled by MQIPT in the same way that it handles the `Sender/Receiver` configuration.

## Channel termination and failure conditions

When MQIPT detects closure (either normal or abnormal) of a IBM MQ channel, it propagates the channel closure. If you close a route by using MQIPT, all channels going through that route are closed.

MQIPT provides an optional idle-timeout facility. If MQIPT detects that a channel has been idle for a period of time exceeding the timeout, it performs an immediate shutdown on the two connections in question.

The IBM MQ systems at either end of the channel observe these abnormal shutdown conditions either as network failures or as termination of the channel by their partner. The channel is then able to restart and recover (if the failure happens during a protocol in-doubt period) as if MQIPT was not being used.

## Safety of messages

When using fast, non-persistent IBM MQ messages, if the MQIPT route fails or is restarted when a IBM MQ message is in transit, the message might be lost. Before restarting the route, make sure that all IBM MQ channels using the MQIPT route are inactive.

See the IBM MQ product documentation for more information about IBM MQ messages and channels.

## Multi-instance queue managers and high availability

MQIPT can be used with multi-instance queue managers in high availability environments.

MQIPT has no persistent state and so there is no benefit in failing over MQIPT to another system. Instead, have multiple instances of MQIPT with identical `mqipt.conf` configuration files running on different systems. Monitor each instance of MQIPT for availability and restart it (on the same system) if necessary. This provides a set of identical MQIPT instances that can be used to route connections. You must then ensure that IBM MQ can route connections to MQIPT and that MQIPT can forward those connections to the destination queue manager.

Outbound IBM MQ channels can be directed to an available MQIPT instance in various ways, for example:

- Use a load balancer or high availability router, such as the IBM Network Dispatcher from the WebSphere® Edge Components product.
- Specify multiple connection names in the IBM MQ channel definition using a comma-separated list. IBM MQ then tries to connect to each MQIPT address in turn until it finds an available MQIPT instance.

You must also direct connections from MQIPT to the destination queue manager. If the high-availability configuration ensures that the IP address fails over with the destination queue manager, then no special MQIPT configuration is required: specify the destination IP address in the **`Destination`** route property and allow the failover operation to move the IP address with the queue manager.

However, if the IP address of the queue manager changes after a failover then you must arrange for MQIPT to forward the connection to the correct destination. This could be done in one of several ways:

- Write a routing exit that checks which IP address and port number are accessible, and then override the route destination for each connection. Some sample routing exits are provided with MQIPT; they can be adapted for this purpose.
- Use a high availability load balancer to redirect the connection.
- Define multiple MQIPT routes, one for each IP address and port where the queue manager might be running. Then direct the IBM MQ connections to the various MQIPT routes, for example by listing all of the route IP addresses and port numbers in a comma-separated list in the connection name of the outbound channel.

It is also important to tune all of the end-to-end components on the network path:

1. Connection attempts to unavailable systems must fail promptly so that reconnect attempts can move on to the first available destination.

   For MQIPT SSL routes, tune the **`SSLClientConnectTimeout`** route property to ensure prompt connection failure for unavailable destinations. Refer to the IBM MQ documentation for details of IBM MQ tuning parameters. Also, consult your operating system documentation for details of TCP/IP tuning for the operating system. In all cases, failed connection attempts should quickly return a network failure (for example, a TCP reset packet), or should time out without undue delay.
2. Active connections to a failed system must be severed promptly so that new connections can be established.

   You should also consider the impact of a failover at a time when connections are actively using MQIPT. It is likely that network connections will be severed during a failover. For client applications, you can use the IBM MQ automatic client reconnection feature to re-establish severed connections. For message channels, you can specify a short retry interval so that the channel reconnects promptly. Consult the IBM MQ documentation for more information about automatic client reconnection and message channel retry configuration.

# Chapter 4. Using MQIPT features

MQIPT supports various features, which are described in this section.

- "HTTP support"
- "SOCKS support" on page 22
- "SSL/TLS support" on page 24
- "Network Dispatcher support" on page 47
- "Java Security Manager" on page 49
- "Security exits" on page 52
- "Port number control" on page 57
- "Other security considerations" on page 57
- "Connection logs" on page 58

## HTTP support

MQIPT supports HTTP tunneling. MQIPT can be configured so that the data packets it forwards are encoded as HTTP requests.

IBM MQ channels do not accept HTTP requests. Therefore a second MQIPT is required to receive the HTTP requests and convert them back into normal IBM MQ protocol packets. The second MQIPT strips off the HTTP header to convert the incoming packet back into a standard IBM MQ protocol packet, before passing it on to the destination queue manager.

When HTTP is being used between two instances of MQIPT, the TCP/IP connection on which the HTTP requests and replies flow is persistent and is kept open for the lifetime of the message channel. MQIPT does not close the TCP/IP connection between request/reply pairs.

If two instances of MQIPT are communicating through HTTP, it is possible that an HTTP request might stay outstanding for an extended period. An example is in a requester/server channel, when the server side is waiting for new messages to arrive on its transmission queue. The IBM MQ channel protocol provides a "heartbeat" mechanism, which requires the waiting end periodically to send heartbeat messages to its partner. The default channel heartbeat period is 5 minutes. MQIPT uses this heartbeat as the HTTP reply. Do not disable this channel heartbeat, or set it to an excessively high value, to avoid causing problems with timeouts in some firewalls.

MQIPT accepts HTTP traffic in chunked format, generated by an HTTP proxy or server.

Some HTTP proxies have their own properties for controlling persistent connections, for example, the number of requests that can be made on a persistent connection. The HTTP proxy must also support HTTP 1.1 protocol. When using the IBM WebSphere Caching Proxy, the following properties should be set:

- **`MaxPersistenceRequest`** set to a high value (for example, `5000`)
- **`PersistentTimeout`** set to a high value (for example, 12 hours)
- **`ProxyPersistence`** set to `on`

To maintain the one-to-one mapping of persistent connections across the Caching Proxy you must specify the **UseSession** field on the Proxy directive. For example, when using a default UriName on MQIPT1 and where MQIPT2 is listening on port address 1415:

```
MQIPT1  -->  Caching Proxy  -->  MQIPT2
```

update the `ibmproxy.conf` file with either:

```
Proxy http://mqipt2:1415/mqipt  http://mqipt2:1415/mqipt  UseSession
```

or

```
Proxy http://mqipt2:*/mqipt  http://mqipt2:*/mqipt  UseSession
```

See "Scenario: Configuring an HTTP proxy" on page 83 for an example of using HTTP.

### HTTPS

HTTPS can be used on an HTTP connection by enabling the HTTPS and SSLClient route properties on the MQIPT issuing the client connection. MQIPT must have access to the trusted CA certificate that will be used to authenticate the target HTTP proxy/server. The SSLClientCAKeyring property can be used to define the key-ring file containing the trusted CA certificate.

A common setup for HTTPS will use a local HTTP proxy to tunnel out through a firewall and connect to a remote HTTP server (or another proxy), which will in turn connect to the remote MQIPT. This MQIPT on the server side of the connection does not need any specific configuration, as the connection request is treated as any normal HTTP connection.

MQIPT uses the HTTPProxy and HTTPServer properties to distinguish the local and remote proxies. HTTPProxy is seen to be the local HTTP proxy and HTTPServer the remote server (or proxy).

HTTPS connections are normally made to listener port address 443 on the HTTP proxy/server, but the HTTPProxyPort and HTTPServerPort can be used to override this default.

## SOCKS support

A SOCKS proxy is a network service used as a controlled point of exit through a firewall. A SOCKS enabled application, running inside the firewall, can use the SOCKS proxy to connect to a remote application.

MQIPT can act as a SOCKS proxy by enabling the **`SocksServer`** property, thereby allowing a SOCKS-enabled IBM MQ application to connect through MQIPT to a remote IBM MQ queue manager. When using this feature, the target destination and destination port address are obtained during the SOCKS handshaking process and therefore the Destination and DestinationPort route properties are overridden. This is a key feature for supporting IBM MQ clustering.

MQIPT can also act as a SOCKS client, on behalf of a local IBM MQ application which has not been SOCKS enabled. This is useful when using a firewall that allows outbound connections only via a SOCKS proxy. Each MQIPT route can be configured to communicate with a different SOCKS proxy.

See “Scenario: Configuring a SOCKS proxy” on page 87 for an example of how to use SOCKS.

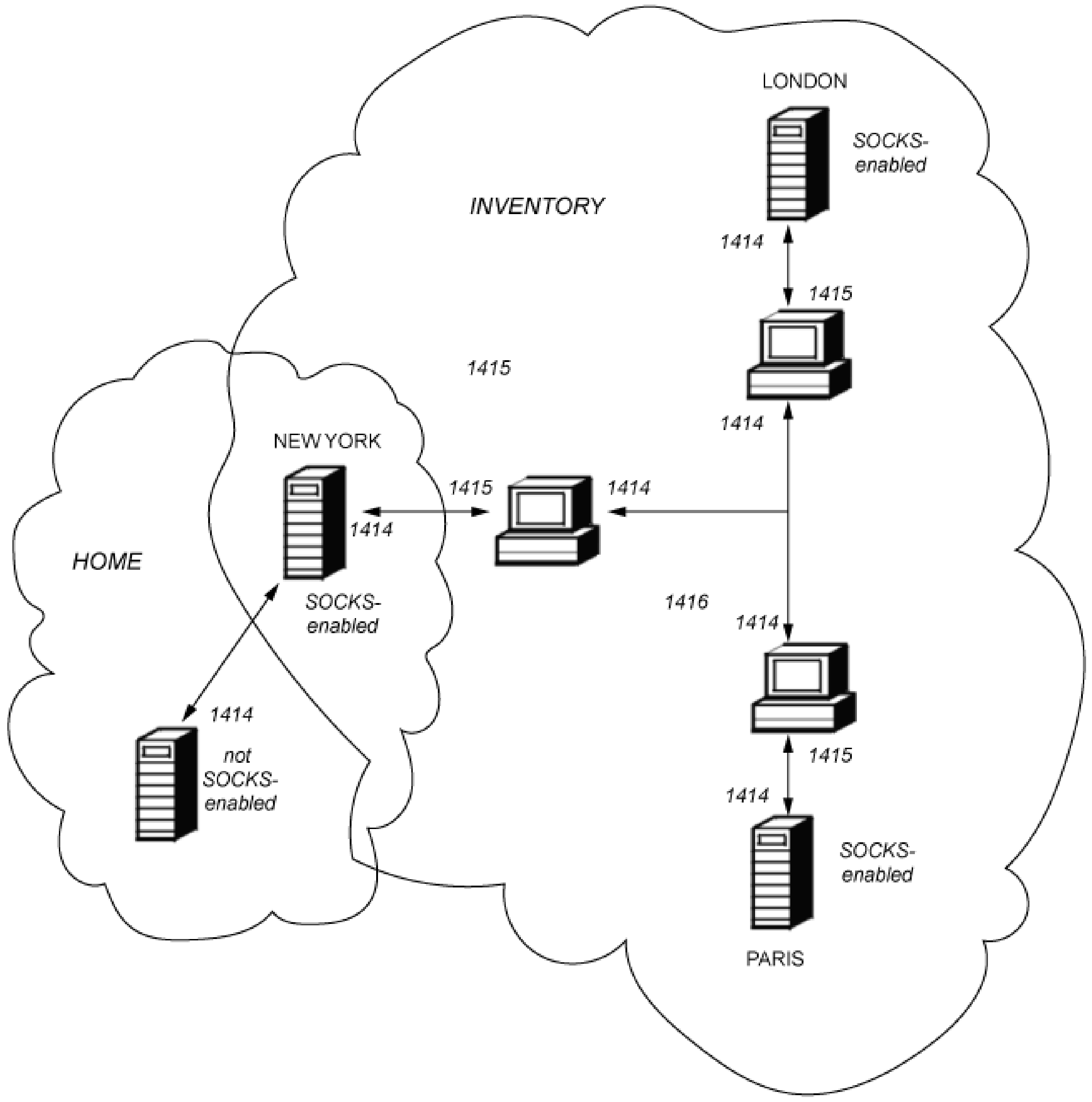
## Clustering

IBM MQ clusters can be used with MQIPT by SOCKS-enabling each queue manager in the cluster that spans the internet and by enabling MQIPT to act as SOCKS proxy. In the following diagram, NEWYORK and CHICAGO are in a cluster called HOME and both hold full repositories. NEWYORK, LONDON and PARIS are in another cluster called INVENTORY. Note that CHICAGO does not need to be SOCKS-enabled as it is in a cluster that does not need an MQIPT.

Each queue manager in the INVENTORY cluster is effectively "hidden" behind an MQIPT. As the queue manager has been SOCKS-enabled, when a cluster-sender channel is started, the request is sent out to its destination, using MQIPT acting as a SOCKS proxy. Normally, the CONNAME on a cluster-receiver channel is used to identify the local queue manager, but when used with MQIPT, the CONNAME must identify the local MQIPT and its incoming listener port. In the following diagram, all incoming listener port addresses are 1414 and the outgoing listener port addresses are 1415.

There are two ways to run a SOCKS-enabled queue manager. The first is to SOCKS-enable the whole computer where the queue manager is running. The second is to SOCKS-enable just the queue manager. Using either method, you must configure the SOCKS client so it only makes remote connections using MQIPT as the SOCKS proxy and disable user authentication. There are a number of products on the market to achieve SOCKS support. You must choose one that supports SOCKS V5 protocol.

See “Scenario: Configuring MQIPT clustering support” on page 90 for an example of how to configure a cluster network.



## SSL/TLS support

Secure sockets can be used to ensure communication privacy, communication integrity, and authentication.

**Communication privacy**
The connection can be made private. The data to be exchanged between the client and the server can be encrypted and only the sender and receiver can make sense of the data. This means that private information, such as credit card numbers, can be transferred securely.

**Communication integrity**
The connection is reliable. The message transport includes a message integrity check based on a secure hash function.

**Authentication**
The client can authenticate the server and an authenticated server can authenticate the client. This means that the information is guaranteed to be exchanged only between the intended parties. The authentication mechanism is based on the exchange of digital certificates (X.509v3 certificates).

## Secure sockets protocols

In MQIPT, secure sockets are provided by using the Secure Sockets Layer (SSL) and the newer Transport Layer Security (TLS) protocols. The two secure sockets protocols are similar but do not interoperate. Both SSL and TLS provide similar security features and in this documentation the terms are used interchangeably unless a specific difference is noted. MQIPT supports SSL version 3.0, TLS 1.0, TLS 1.1, and TLS 1.2 provided by the supplied Java Runtime Environment (JRE). The IBM MQ CipherSpec of the remote channel determines which protocol MQIPT uses. SSL version 3.0 is insecure and so is disabled by default from version 2.1.0.2 of MQIPT. If you need to use SSL, it can be reenabled by specifying `SSLv3` in the **`SSLServerProtocols`** and **`SSLClientProtocols`** route properties.

The SSL/TLS protocols can use different digital signature algorithms for authentication of communication parties. The cryptographic operations that are used in SSL/TLS, encryption for data confidentiality, and secure hashing for message integrity, rely on the sharing of secret keys between the client and the server. SSL/TLS provides various key exchange mechanisms that allow for the sharing of secret keys. SSL/TLS can make use of various algorithms for encryption and hashing.

## JRE cryptographic component

The SSL/TLS cryptographic component of the JRE contains the IBMJSSEFIPS and IBMJCEFIPS security providers, which are certified compliant with FIPS 140-2 at level 1. These security providers have the highest priority in the JRE so that FIPS-certified implementations are used wherever available. Various cryptographic algorithms are supported; specify them by using SSL/TLS CipherSuites. Not all CipherSuites are FIPS 140-2 certified.

## SSL/TLS Proxy Mode

As an alternative to using SSL/TLS secure sockets directly, an MQIPT route can be configured to operate in SSL/TLS Proxy Mode. In this mode, the route only forwards SSL/TLS data between the two IBM MQ end-points; it does not participate in the SSL/TLS handshake and does not require any digital certificates.

You can use SSL/TLS Proxy Mode in cases where the IBM MQ channels that communicate through MQIPT are already configured for SSL/TLS communication and you want to use MQIPT for another purpose, such as routing connections through firewalls or restricting the set of allowable connections via a security exit. When running in SSL/TLS Proxy Mode, MQIPT checks that the initial SSL/TLS packets are valid before passing on a new connection to the destination.

IBM MQ version 8.0 supports the use of multiple certificates on the same queue manager, using a per-channel certificate label attribute. Inbound channels to the queue manager (for example, server connection or receiver) rely on detecting the channel name using TLS Server Name Indication, in order to present the correct certificate from the queue manager.

If you use MQIPT with a route that has both *SSLServer* and *SSLClient* set, there are two separate TLS sessions between the endpoints, and the SNI data will not flow across the session break.

You can use separate MQIPT routes to get multiple certificate support by selecting the appropriate certificate, for example through the *SSLServerSiteLabel* and *SSLClientSiteLabel* route properties. Alternatively, use MQIPT *SSLProxyMode* which forwards all SSL or TLS control flows intact, including the SNI name.

Note that multiple certificates for inbound channels with a certificate label across MQIPT work only if you are using SSL/TLS proxy mode.

## CipherSuites supported by MQIPT

The following table shows which CipherSuites are supported by MQIPT and which are enabled by default.

By default, only a subset of CipherSuites are enabled. Use the **SSLClientCipherSuites** and **SSLServerCipherSuites** route properties if you want to override the default set of enabled CipherSuites.

The following CipherSuite algorithms are reported as insecure: RC4, DHE, and DH. CipherSuites based on these algorithms are no longer supported from MQIPT version 2.1.0.2. 3DES ciphers are no longer considered secure and are disabled by default from MQIPT version 2.1.0.3. If you are aware of the potential hazards but still need to use one of these CipherSuites, you can add support for it to MQIPT. See Note 5 after the table.

| CipherSuite | Enabled by default (Note 4) | V2.1.0.3 support (Note 5) |
|---|---|---|
| SSL_DH_anon_EXPORT_WITH_DES40_CBC_SHA | | |
| SSL_DH_anon_EXPORT_WITH_RC4_40_MD5 | | |
| SSL_DH_anon_WITH_3DES_EDE_CBC_SHA | | |
| SSL_DH_anon_WITH_AES_128_CBC_SHA | | |
| SSL_DH_anon_WITH_AES_128_CBC_SHA256 | | |
| SSL_DH_anon_WITH_AES_256_CBC_SHA | | |
| SSL_DH_anon_WITH_AES_256_CBC_SHA256 | | |
| SSL_DH_anon_WITH_DES_CBC_SHA | | |
| SSL_DH_anon_WITH_RC4_128_MD5 | | |
| SSL_DHE_DSS_EXPORT_WITH_DES40_CBC_SHA | | |
| SSL_DHE_DSS_WITH_3DES_EDE_CBC_SHA | Yes (Note 2) | |
| SSL_DHE_DSS_WITH_AES_128_CBC_SHA | Yes (Note 2) | |
| SSL_DHE_DSS_WITH_AES_128_CBC_SHA256 | Yes (Note 2) | |
| SSL_DHE_DSS_WITH_AES_128_GCM_SHA256 | | |

| CipherSuite | Enabled by default (Note 4) | V2.1.0.3 support (Note 5) |
|---|---|---|
| SSL_DHE_DSS_WITH_AES_256_CBC_SHA | Yes (Note 2) | |
| SSL_DHE_DSS_WITH_AES_256_CBC_SHA256 | Yes (Note 2) | |
| SSL_DHE_DSS_WITH_DES_CBC_SHA | | |
| SSL_DHE_DSS_WITH_RC4_128_SHA | | |
| SSL_DHE_RSA_EXPORT_WITH_DES40_CBC_SHA | | |
| SSL_DHE_RSA_WITH_3DES_EDE_CBC_SHA | Yes (Note 2) | |
| SSL_DHE_RSA_WITH_AES_128_CBC_SHA | Yes (Note 2) | |
| SSL_DHE_RSA_WITH_AES_128_CBC_SHA256 | Yes (Note 2) | |
| SSL_DHE_RSA_WITH_AES_128_GCM_SHA256 | | |
| SSL_DHE_RSA_WITH_AES_256_CBC_SHA | Yes (Note 2) | |
| SSL_DHE_RSA_WITH_AES_256_CBC_SHA256 | Yes (Note 2) | |
| SSL_DHE_RSA_WITH_DES_CBC_SHA | | |
| SSL_ECDH_anon_WITH_3DES_EDE_CBC_SHA | | |
| SSL_ECDH_anon_WITH_AES_128_CBC_SHA | | Yes |
| SSL_ECDH_anon_WITH_AES_256_CBC_SHA | | Yes |
| SSL_ECDH_anon_WITH_NULL_SHA | | Yes |
| SSL_ECDH_anon_WITH_RC4_128_SHA | | |
| SSL_ECDH_ECDSA_WITH_3DES_EDE_CBC_SHA | Yes (Note 6) | |
| SSL_ECDH_ECDSA_WITH_AES_128_CBC_SHA | Yes | Yes |
| SSL_ECDH_ECDSA_WITH_AES_128_CBC_SHA256 | Yes | Yes |
| SSL_ECDH_ECDSA_WITH_AES_128_GCM_SHA256 | | Yes |
| SSL_ECDH_ECDSA_WITH_AES_256_CBC_SHA | Yes | Yes |
| SSL_ECDH_ECDSA_WITH_AES_256_CBC_SHA384 | Yes | Yes |
| SSL_ECDH_ECDSA_WITH_NULL_SHA | | Yes |
| SSL_ECDH_ECDSA_WITH_RC4_128_SHA | Yes (Note 3) | |
| SSL_ECDH_RSA_WITH_3DES_EDE_CBC_SHA | Yes (Note 6) | |
| SSL_ECDH_RSA_WITH_AES_128_CBC_SHA | Yes | Yes |
| SSL_ECDH_RSA_WITH_AES_128_CBC_SHA256 | Yes | Yes |
| SSL_ECDH_RSA_WITH_AES_128_GCM_SHA256 | | Yes |
| SSL_ECDH_RSA_WITH_AES_256_CBC_SHA | Yes | Yes |
| SSL_ECDH_RSA_WITH_AES_256_CBC_SHA384 | Yes | Yes |
| SSL_ECDH_RSA_WITH_NULL_SHA | | Yes |
| SSL_ECDH_RSA_WITH_RC4_128_SHA | Yes (Note 3) | |
| SSL_ECDHE_ECDSA_WITH_3DES_EDE_CBC_SHA | Yes (Note 6) | |
| SSL_ECDHE_ECDSA_WITH_AES_128_CBC_SHA | Yes | Yes |
| SSL_ECDHE_ECDSA_WITH_AES_128_CBC_SHA256 | Yes | Yes |
| SSL_ECDHE_ECDSA_WITH_AES_128_GCM_SHA256 | | Yes |
| SSL_ECDHE_ECDSA_WITH_AES_256_CBC_SHA | Yes | Yes |
| SSL_ECDHE_ECDSA_WITH_AES_256_CBC_SHA384 | Yes | Yes |

| CipherSuite | Enabled by default (Note 4) | V2.1.0.3 support (Note 5) |
|---|---|---|
| SSL_ECDHE_ECDSA_WITH_NULL_SHA | | Yes |
| SSL_ECDHE_ECDSA_WITH_RC4_128_SHA | Yes (Note 3) | |
| SSL_ECDHE_RSA_WITH_3DES_EDE_CBC_SHA | Yes (Note 6) | |
| SSL_ECDHE_RSA_WITH_AES_128_CBC_SHA | Yes | Yes |
| SSL_ECDHE_RSA_WITH_AES_128_CBC_SHA256 | Yes | Yes |
| SSL_ECDHE_RSA_WITH_AES_128_GCM_SHA256 | | Yes |
| SSL_ECDHE_RSA_WITH_AES_256_CBC_SHA | Yes | Yes |
| SSL_ECDHE_RSA_WITH_AES_256_CBC_SHA384 | Yes | Yes |
| SSL_ECDHE_RSA_WITH_NULL_SHA | | Yes |
| SSL_ECDHE_RSA_WITH_RC4_128_SHA | Yes (Note 3) | |
| SSL_KRB5_EXPORT_WITH_DES_CBC_40_MD5 | | Yes |
| SSL_KRB5_EXPORT_WITH_DES_CBC_40_SHA | | Yes |
| SSL_KRB5_EXPORT_WITH_RC4_40_MD5 | | |
| SSL_KRB5_EXPORT_WITH_RC4_40_SHA | | |
| SSL_KRB5_WITH_3DES_EDE_CBC_MD5 | | |
| SSL_KRB5_WITH_3DES_EDE_CBC_SHA | | |
| SSL_KRB5_WITH_DES_CBC_MD5 | | Yes |
| SSL_KRB5_WITH_DES_CBC_SHA | | Yes |
| SSL_KRB5_WITH_RC4_128_MD5 | | |
| SSL_KRB5_WITH_RC4_128_SHA | | |
| SSL_RSA_EXPORT_WITH_DES40_CBC_SHA | | Yes |
| SSL_RSA_EXPORT_WITH_RC4_40_MD5 | | |
| SSL_RSA_FIPS_WITH_3DES_EDE_CBC_SHA (Note 1) | | |
| SSL_RSA_FIPS_WITH_DES_CBC_SHA (Note 1) | | Yes |
| SSL_RSA_WITH_3DES_EDE_CBC_SHA | Yes (Note 6) | |
| SSL_RSA_WITH_AES_128_CBC_SHA | Yes | Yes |
| SSL_RSA_WITH_AES_128_CBC_SHA256 | Yes | Yes |
| SSL_RSA_WITH_AES_128_GCM_SHA256 | | Yes |
| SSL_RSA_WITH_AES_256_CBC_SHA | Yes | Yes |
| SSL_RSA_WITH_AES_256_CBC_SHA256 | Yes | Yes |
| SSL_RSA_WITH_DES_CBC_SHA | | Yes |
| SSL_RSA_WITH_NULL_MD5 | | Yes |
| SSL_RSA_WITH_NULL_SHA | | Yes |
| SSL_RSA_WITH_NULL_SHA256 | | Yes |
| SSL_RSA_WITH_RC4_128_MD5 | Yes (Note 3) | |
| SSL_RSA_WITH_RC4_128_SHA | Yes (Note 3) | |

**Notes:**

1. Although this CipherSuite is supported for compatibility with previous versions, it is no longer FIPS-compliant and its use should be avoided.

2. From version 2.1.0.2, if you want these CipherSuites to be enabled as defaults, remove DHE from the `jdk.tls.disabledAlgorithms` list as described in Note 5.
3. From version 2.1.0.2, if you want these CipherSuites to be enabled as defaults, remove RC4 from the `jdk.tls.disabledAlgorithms` list as described in Note 5.
4. The second column shows which CipherSuites are enabled as defaults. However, they will only be used if the corresponding algorithm is supported, as indicated in the third column.
5. The third column shows which CipherSuites are supported in version 2.1.0.3 and later. You can add support for CipherSuites that are not normally supported by removing the corresponding algorithm (RC4, DHE, DH, 3DES, or DES) from the list of disabled algorithms (`jdk.tls.disabledAlgorithms`) in the `java.security` file, found in *`mqipt_path`*`/java/jre/lib/security/`, where *`mqipt_path`* is the location where MQIPT is installed.
6. From version 2.1.0.3, if you want these CipherSuites to be enabled as defaults, remove 3DES and DESede from the `jdk.tls.disabledAlgorithms` list as described in Note 5.

## IBM MQ CipherSpecs and MQIPT CipherSuites

The following table shows the relationship between the CipherSpecs supported by IBM MQ and the CipherSuites supported by MQIPT.

The table also shows the protocol version that IBM MQ expects each CipherSpec to use.

An IBM MQ CipherSpec uniquely determines both the encryption algorithm and also the secure socket protocol version to be used. Some IBM MQ CipherSpecs differ only by protocol version, so it is not sufficient to configure the CipherSuite alone. The SSL/TLS handshake negotiates the highest secure sockets protocol version supported by both sides, and then selects a CipherSuite from the set of mutually enabled ciphers.

For example, an SSLClient route with `SSLClientCipherSuites=SSL_RSA_WITH_3DES_EDE_CBC_SHA` could negotiate either TLS_RSA_WITH_3DES_EDE_CBC_SHA (TLS 1.0) or TRIPLE_DES_SHA_US (SSL 3.0) with the remote queue manager. In fact it is possible to negotiate this CipherSuite over TLS 1.2, but IBM MQ does not support this CipherSuite over TLS 1.2. For this reason, SSLClient routes are particularly likely to cause AMQ9616 or AMQ9631 errors at the queue manager.

To avoid such errors on SSLClient routes, set the **`SSLClientProtocols`** route property to the appropriate value for the intended CipherSpec. In some cases it might also be necessary to restrict the server-side protocol set by using the **`SSLServerProtocols`** route property. Use the protocol version shown in the table to determine the correct setting for these route properties.

This issue particularly affects the following CipherSuites and CipherSpecs for SSLClient routes:
- SSL_RSA_WITH_3DES_EDE_CBC_SHA, which corresponds to:
  - SSL 3.0: MQ CipherSpec TRIPLE_DES_SHA_US
  - TLS 1.0: MQ CipherSpec TLS_RSA_WITH_3DES_EDE_CBC_SHA
- SSL_RSA_WITH_DES_CBC_SHA, which corresponds to:
  - SSL 3.0: MQ CipherSpec DES_SHA_EXPORT

  - TLS 1.0: MQ CipherSpec TLS_RSA_WITH_DES_CBC_SHA
- SSL_RSA_WITH_RC4_128_SHA, which corresponds to:
  - SSL 3.0: MQ CipherSpec RC4_SHA_US
  - TLS 1.0: MQ CipherSpec TLS_RSA_WITH_RC4_128_SHA256

If you want to use a single MQIPT SSLClient route to tunnel multiple IBM MQ channels that use different CipherSpecs, ensure that all channels have CipherSpecs that use the same secure sockets protocol version as each other and that you set **SSLClientProtocols** to use this single protocol version.

For more information about IBM MQ CipherSpecs, see *Enabling CipherSpecs* in the IBM MQ Version 9.0 Knowledge Center.

| IBM MQ CipherSpec | MQIPT CipherSuite | Protocol version |
|---|---|---|
| DES_SHA_EXPORT | SSL_RSA_WITH_DES_CBC_SHA | SSLv3 |
| DES_SHA_EXPORT1024 | N/A | N/A |
| ECDHE_ECDSA_3DES_EDE_CBC_SHA256 | SSL_ECDHE_ECDSA_WITH_3DES_EDE_CBC_SHA | TLSv1.2 |
| ECDHE_ECDSA_AES_128_CBC_SHA256 | SSL_ECDHE_ECDSA_WITH_AES_128_CBC_SHA256 | TLSv1.2 |
| ECDHE_ECDSA_AES_128_GCM_SHA256 | SSL_ECDHE_ECDSA_WITH_AES_128_GCM_SHA256 | TLSv1.2 |
| ECDHE_ECDSA_AES_256_CBC_SHA384 | SSL_ECDHE_ECDSA_WITH_AES_256_CBC_SHA384 | TLSv1.2 |
| ECDHE_ECDSA_AES_256_GCM_SHA384 | N/A | N/A |
| ECDHE_ECDSA_NULL_SHA256 | SSL_ECDHE_ECDSA_WITH_NULL_SHA | TLSv1.2 |
| ECDHE_ECDSA_RC4_128_SHA256 | SSL_ECDHE_ECDSA_WITH_RC4_128_SHA | TLSv1.2 |
| ECDHE_RSA_3DES_EDE_CBC_SHA256 | SSL_ECDHE_RSA_WITH_3DES_EDE_CBC_SHA | TLSv1.2 |
| ECDHE_RSA_AES_128_CBC_SHA256 | SSL_ECDHE_RSA_WITH_AES_128_CBC_SHA256 | TLSv1.2 |
| ECDHE_RSA_AES_128_GCM_SHA256 | SSL_ECDHE_RSA_WITH_AES_128_GCM_SHA256 | TLSv1.2 |
| ECDHE_RSA_AES_256_CBC_SHA384 | SSL_ECDHE_RSA_WITH_AES_256_CBC_SHA384 | TLSv1.2 |
| ECDHE_RSA_AES_256_GCM_SHA384 | N/A | N/A |
| ECDHE_RSA_NULL_SHA256 | SSL_ECDHE_RSA_WITH_NULL_SHA | TLSv1.2 |
| ECDHE_RSA_RC4_128_SHA256 | SSL_ECDHE_RSA_WITH_RC4_128_SHA | TLSv1.2 |
| FIPS_WITH_3DES_EDE_CBC_SHA (Note 1) | SSL_RSA_FIPS_WITH_3DES_EDE_CBC_SHA | SSLv3 |
| FIPS_WITH_DES_CBC_SHA (Note 1) | SSL_RSA_FIPS_WITH_DES_CBC_SHA | SSLv3 |
| NULL_MD5 | SSL_RSA_WITH_NULL_MD5 | SSLv3 |
| NULL_SHA | SSL_RSA_WITH_NULL_SHA | SSLv3 |
| RC2_MD5_EXPORT | N/A | N/A |
| RC4_56_SHA_EXPORT1024 | N/A | N/A |
| RC4_MD5_EXPORT | SSL_RSA_EXPORT_WITH_RC4_40_MD5 | SSLv3 |
| RC4_MD5_US | SSL_RSA_WITH_RC4_128_MD5 | SSLv3 |
| RC4_SHA_US | SSL_RSA_WITH_RC4_128_SHA | SSLv3 |
| TLS_RSA_WITH_3DES_EDE_CBC_SHA | SSL_RSA_WITH_3DES_EDE_CBC_SHA | TLSv1 |
| TLS_RSA_WITH_AES_128_CBC_SHA | SSL_RSA_WITH_AES_128_CBC_SHA | TLSv1 |
| TLS_RSA_WITH_AES_128_CBC_SHA256 | SSL_RSA_WITH_AES_128_CBC_SHA256 | TLSv1.2 |
| TLS_RSA_WITH_AES_128_GCM_SHA256 | SSL_RSA_WITH_AES_128_GCM_SHA256 | TLSv1.2 |
| TLS_RSA_WITH_AES_256_CBC_SHA | SSL_RSA_WITH_AES_256_CBC_SHA | TLSv1 |

| IBM MQ CipherSpec | MQIPT CipherSuite | Protocol version |
|---|---|---|
| TLS_RSA_WITH_AES_256_CBC_SHA256 | SSL_RSA_WITH_AES_256_CBC_SHA256 | TLSv1.2 |
| TLS_RSA_WITH_AES_256_GCM_SHA384 | N/A | N/A |
| TLS_RSA_WITH_DES_CBC_SHA | SSL_RSA_WITH_DES_CBC_SHA | TLSv1 |
| TLS_RSA_WITH_NULL_NULL | N/A | N/A |
| TLS_RSA_WITH_NULL_SHA256 | SSL_RSA_WITH_NULL_SHA256 | TLSv1.2 |
| TLS_RSA_WITH_RC4_128_SHA256 | SSL_RSA_WITH_RC4_128_SHA | TLSv1.2 |
| TRIPLE_DES_SHA_US | SSL_RSA_WITH_3DES_EDE_CBC_SHA | SSLv3 |

**Note:**

1. Although this CipherSuite is supported for compatibility with previous versions, it is no longer FIPS-compliant and its use should be avoided.

## SSL/TLS handshake

The SSL/TLS handshaking process occurs during the initial connection request between the SSL/TLS client and server, when authentication and negotiation of CipherSuites is performed.

All the supported SSL/TLS CipherSuites (see “SSL/TLS support” on page 24), with the exception of the anonymous CipherSuites, require server authentication and allow client authentication; the server can be configured to request client authentication. You should avoid using anonymous CipherSuites because they provide no guarantees about the identity of the remote peer. It is possible for a man-in-the-middle attack to intercept anonymous SSL/TLS connections without your knowledge. Use anonymous CipherSuites only on trustworthy internal networks and only if you are prepared to accept the risk of data interception.

The communication peer authentication in SSL/TLS is based on public key cryptography and X.509v3 digital certificates. A site that should be authenticated in the SSL/TLS protocol requires a private key and a digital certificate (which contains the corresponding public key together with the information about the site's identity), validity time of the certificate. The certificates are signed by a Certification Authority, the certificates of such authorities are called signer certificates. A certificate followed by one or more signer certificates constitute a certificate chain. A certificate chain is characterized by the fact that, starting from the first certificate (site certificate), the signature of each certificate in the chain can be verified using the public key contained in the next signer certificate.

When a secure connection requiring server authentication is being established the server sends to the client a certificate chain to prove its identity. The SSL/TLS client will pursue the connection establishment to the server only if it can authenticate the server, for example, verify the signature of the server's site certificate. In order to verify that signature the SSL/TLS client needs to trust the server site itself or at least one of the signers in the certificate chain provided by the server. The certificates of the trusted sites and signers must be maintained on the client side to perform this verification.

The SSL/TLS client inspects the server's certificate chain, starting with the site certificate. The client considers the signature of the site certificate to be valid in the following circumstances:

- The site certificate is in the repository of the trusted site or signer certificates

- A signer certificate in the chain can be validated based on its repository of trusted signer certificates

In the latter case, the SSL/TLS client checks that the certificate chain is indeed correctly signed, from the trusted signer certificate down to the server's site certificate. Each certificate involved in this process is also examined for correctness of format and dates of validity. If any of these checks fail, the connection to the server is refused. After verifying the server certificate the client uses the public key embedded in that certificate in the next steps of the SSL/TLS protocol. The SSL/TLS connection can only be established if the server really has the corresponding private key.

The client authentication follows the same procedure: if a SSL/TLS server requires client authentication the client sends to the server a certificate chain to prove its identity. The server verifies the chain based on its repository of trusted site and signer certificates. After verifying the client's certificate the server uses the public key embedded in that certificate in the next steps of the SSL/TLS protocol. The SSL/TLS connection can only be established if the client really has the corresponding private key.

The SSL and TLS protocols provide high security communications (although TLS is considered to be more secure than SSL). However, the protocol operates based on the information provided by the application. Only if that information base is also maintained securely the overall goal of secure communication can be achieved. For example, if your repository of trusted site and signer certificates is compromised, you might establish a secure connection to a very insecure communication partner.

## MQIPT implementation of SSL/TLS

SSL version 3.0 and TLS versions 1.0, 1.1, and 1.2 are implemented with Public Key Cryptography Standards (PKCS) #12 tokens stored in key-ring files (with file types of `.p12` or `.pfx`), containing X509.V3 certificates.

A key-ring file can also contain Certificate Revocation Lists (CRLs) and Authority Revocation Lists (ARLs).

MQIPT uses the IBM Java Secure Sockets Extension (JSSE) package.

MQIPT can act as an SSL/TLS client or an SSL/TLS server depending on which end initiates the connection. The client starts a connection and the server accepts the connection request. It is possible for an MQIPT route to act both as a client and a server, In this case, using the SSL/TLS Proxy Mode feature typically gives better performance.

When MQIPT is configured for SSL/TLS Proxy Mode, it only forwards SSL/TLS data between the two end-points; it does not participate in the SSL/TLS handshake and does not require any digital certificates.

Each MQIPT route can be independently configured with its own set of SSL/TLS properties. See "Route properties" on page 125 for more details.

## Encrypting a keyring password

The password used to open a keyring file can be encrypted with the **`mqiptPW`** script. The encrypted password is stored in a file, which can be used by any of the following properties: **`SSLClientKeyRingPW`**, **`SSLClientCAKeyRingPW`**,

**SSLServerKeyRingPW**, and **SSLServerCAKeyRingPW**. This topic describes the correct way to store a key-ring password for use by MQIPT.

The iKeyman stash file facility is not supported by MQIPT. Instead of using a stash file, you must use the **mqiptPW** command to store the encrypted password.

Command format:

```
mqiptPW password filename -replace
```

where

***password***
: is the clear text password needed to open the given keyring file

***filename***
: is the name of the password file to be created

The `-replace` option overwrites an existing password file with the same name.

Passwords can include the space character, but the whole password string must be enclosed in quotes for this to be acceptable. There is no limit to the length or format of a password.

**Note:** if you have migrated from a previous level of MQIPT, you must replace the current password files containing the clear-text password with a copy of the encrypted password file.

You must use the password `mqiptSample` to open either of the sample keyring files by using a key management utility.

## Selecting certificates from a key-ring file

It is possible to have more than one personal certificate stored in the same key-ring file, so the **SSLClientSite*** properties can be used on the client side to select the certificate to be sent to the server for authentication and the **SSLServerSite*** properties can be used on the server side to select the certificate to be sent to the client for authentication.

Using these properties, a certificate can be selected based on its Distinguished Name (DN). Alternatively, the certificate label can be used to select a certificate using the **SSLServerSiteLabel** and **SSLClientSiteLabel** properties.

## Trust settings

A key-ring file contains a personal certificate that includes the signer certificate or chain of signer certificates.

There are two types of key-ring file:

**Certificate Authority (CA) key-ring file**
: This file contains trusted CA certificates that are used to validate certificates belonging to a remote peer. These CA certificates help to determine whether the remote peer is trustworthy. The MQIPT CA key-ring files are identified by the **SSLClientCAKeyRing** and **SSLServerCAKeyRing** route properties.

**Personal certificate key-ring file**
: This file contains personal certificates that MQIPT uses to identify itself to a remote peer. When you generate a self-signed certificate or request a

CA-signed certificate, you should do so by using the personal certificate key-ring file. In MQIPT, personal certificate key-ring files are identified by the **SSLClientKeyRing** and **SSLServerKeyRing** route properties.

The key-ring file on the SSL/TLS server side, identified by the **SSLServerKeyRing** property, should contain its personal certificate.

The key-ring file on the SSL/TLS client side, identified by the **SSLClientCAKeyRing** property should contain a list of trusted CA certificates that will be used to authenticate the certificate sent from the server.

If you also need client authentication, you must enable the **SSLServerAskClientAuth** property on the server side. The key-ring file on the client side, identified by the **SSLClientKeyRing** property, should contain its personal certificate. The key-ring file on the server side, identified by the **SSLServerCAKeyRing** property, should contain a list of trusted CA certificates that will be used to authenticate the client. You should also enable the **SSLServerAskClientAuth** property on all SSLServer routes.

As an alternative to using certificates signed by a trusted CA, you can use self-signed certificates. Examples of these can be found in the sample key-ring files provided with MQIPT in the `ssl` subdirectory: `sslSample.pfx` and `sslCAdefault.pfx`. To open either of these PKCS#12 key-ring files, you must use the password: `mqiptSample`.

Self-signed certificates can be useful in test scenarios where you must ensure SSL/TLS connectivity without paying a CA for a certificate. However, you should not use self-signed certificates in production environments. To create a CA-signed certificate, see "Scenario: Creating a key-ring file" on page 72.

You can use a utility called iKeyman, which is provided with MQIPT, to manage digital certificates and key-ring files. See "iKeyman" on page 37 for installation instructions and further information.

You must protect any key-ring files and password files by using the security features of the operating system to prevent unauthorized access to them.

## Testing SSL/TLS

You can test an SSL/TLS connection by using the examples provided in this documentation.

See Chapter 6, "Scenarios: Getting started with MQIPT," on page 69 for a description of various scenarios. In particular, see the following tasks:

- "Scenario: Authenticating an SSL/TLS server" on page 76
- "Scenario: Authenticating an SSL/TLS client" on page 79
- "Scenario: running MQIPT in SSL/TLS proxy mode" on page 97
- "Scenario: running MQIPT in SSL/TLS proxy mode with a security manager" on page 99

To test that your SSL/TLS configuration works correctly, you can use self-signed certificates. Self-signed certificates are useful in test scenarios so that you can ensure SSL/TLS connectivity without paying a Certificate Authority (CA) for a certificate. See "Scenario: Creating test certificates" on page 75 for details.

You can find examples of self-signed certificates in the sample key-ring files, named `sslSample.pfx` and `sslCAdefault.pfx`, provided with MQIPT in the `ssl` subdirectory. To open either of these PKCS#12 key-ring files, you must use the password `mqiptSample`. These sample certificates are provided for your convenience during testing. However, the private keys of the sample certificates are known to all MQIPT users. This means that they are insecure and should be used only in a test environment.

You should not use any self-signed certificates in production environments, whether they are sample certificates or not. Instead, obtain a CA-signed certificate from a trusted CA. To create a CA-signed certificate, see "Scenario: Creating a key-ring file" on page 72.

When creating or requesting a certificate, you should consider which key type, key size and digital signature algorithm are appropriate for your security needs. See "Digital certificate considerations for MQIPT" on page 38 for further information.

Certificates and certificate management technologies are available from a number of third-party suppliers.

## SSL/TLS error messages

Handshake failures are logged in the MQIPT connection log in the form of JSSE exceptions. See "Connection logs" on page 58. The following table describes the different exceptions, the likely cause and the corresponding action to resolve the failure.

Certificate exceptions usually relate to the certificates at the remote end of the connection.

Where the error relates to the certificate of a IBM MQ client or queue manager, the term *keyring file* includes the IBM MQ key repository of the remote partner.

In MQIPT, CA certificates are stored in the CA keyring file, which is identified by the **`SSLClientCAKeyRing`** and **`SSLServerCAKeyRing`** route properties. If the CA keyring route properties are not set, the corresponding personal keyring file (`SSLClientKeyRing` or `SSLServerKeyRing`) is searched for CA certificates instead.

| Exception | Cause | Action |
|---|---|---|
| CertificateException | The certificate is not trusted because it is signed by a CA that is not in the CA keyring. | Check that all of the necessary CA certificates are present in the CA keyring file. Use the IBM Key Management tool supplied with MQIPT to add any missing CA certificates, taking care to obtain a copy of each CA certificate from a trustworthy source. |
| CertificateExpiredException | 1. The certificate has expired: its **`notAfter`** date has passed.<br>2. The system clock is set incorrectly. | 1. Obtain a new certificate and insert it into the keyring file. If the certificate belongs to a Certificate Authority, place the new certificate into the CA keyring file.<br>2. Check that the UTC system clock is set to the correct time. |

| Exception | Cause | Action |
|---|---|---|
| CertificateNotYetValidException | 1. The certificate is being used prematurely: its **`notBefore`** date has not yet arrived.<br>2. The system clock is set incorrectly. | 1. Check that the certificate has been generated and signed correctly. If your organization operates its own CA, the UTC system clock for the CA might be incorrect.<br>2. Check that the UTC system clock is set to the correct time. |
| CertificateParsingException | 1. The certificate contains invalid DER data.<br>2. The certificate uses unsupported DER features. | Ensure the certificate has been correctly generated and can be viewed in the IBM Key Management tool supplied with MQIPT. Consider obtaining a new certificate with fewer certificate extensions. |
| CertificateRevokedException | Certificate revocation checking is enabled and the certificate was found to be revoked. | The certificate in question should not be trusted. Obtain a replacement certificate and ensure the new certificate and its private key are present in the keyring file. |
| CertPathBuilderException | The certificate chain was not signed by a recognised Certificate Authority. | 1. If you are using CA-signed certificates, check that all root CA and intermediate CA certificates are present in the CA keyring file.<br>2. If you are using self-signed certificates, ensure that you have extracted a copy of the public part of the remote certificate and added it to the CA keyring file. Avoid using self-signed certificates in production environments. |
| CertStoreException<br>KeyStoreException | An error occurred reading a certificate from a keyring for one of the following reasons:<br>1. The keyring file is damaged.<br>2. The keyring file is missing.<br>3. The stored password does not match the keyring file password. | 1. Ensure that the keyring file can be read and that all certificates can be viewed with the IBM Key Management tool.<br>2. Check that all keyring route properties refer to the correct file name.<br>3. Check that the stored keyring file password is correct. Use the mqiptPW tool to store the correct password. |
| SSLException: No available certificate or key corresponds to the SSL cipher suites which are enabled. | You must have a personal certificate with the correct type of key for the CipherSuites you are using. For example, CipherSuites whose names begin with `SSL_ECDH_ECDSA_` require a certificate with an Elliptic Curve public key. The most commonly used CipherSuites require a certificate with an RSA public key. | Open the keyring file with the IBM Key Management tool. Under the Personal Certificates view, select each certificate in turn and view it. Click **View Details** and navigate to the Subject Public Key section to see the public key type. Then check the MQIPT **`SSLClientCipherSuites`** and **`SSLServerCipherSuites`** route properties to ensure that the appropriate CipherSuites are enabled. |

| Exception | Cause | Action |
|---|---|---|
| SSLException: No cipher suites in common. | The handshake has failed to agree a CipherSuite because there is no overlap between the sets of enabled CipherSuites at both ends of the connection. In particular, an outbound IBM MQ connection only enables a single cipher so SSLServer MQIPT routes are particularly likely to experience this error. | Check the list of enabled CipherSuites in the MQIPT **SSLClientCipherSuites** and **SSLServerCipherSuites** route properties. Consider enabling additional CipherSuites. Consult the table provided to determine the correct CipherSuites to enable for each IBM MQ channel CipherSpec value. |

## iKeyman

iKeyman is a certificate and key management application that is already familiar to IBM MQ users. iKeyman can be used to manage symmetric and asymmetric keys, digital certificates, and certificate requests in various different types of key-ring file. It can also be used to manage the key-ring files themselves.

iKeyman uses the term *key database* to refer to a key-ring file; these terms are synonymous.

iKeyman can be run in two modes, graphical user interface (GUI) and command-line interface (CLI). Use the **mqiptKeyman** command to start the iKeyman GUI and the **mqiptKeycmd** command to run the iKeyman CLI.

### Required key-ring file format for MQIPT

When using iKeyman to create key-ring files for use in MQIPT, you must use the PKCS#12 file format:

- In the iKeyman GUI, select `PKCS#12` in the **Key database type** field when creating the key-ring file.
- In the iKeyman CLI, include the `-type pkcs12` parameter on the `mqiptKeycmd -keydb -create` command.

### Encrypting the key-ring password for MQIPT

After creating the key-ring file, you must store the key-ring password in an encrypted file which MQIPT can use to access the file. See “Encrypting a keyring password” on page 32 for information about this.

Note that the iKeyman stash file facility is not supported by MQIPT. You must use the **mqiptPW** command to store the encrypted password instead of using a stash file.

### Command line examples

The iKeyman CLI uses the same syntax as the IBM MQ **runmqckm** command. Append the required parameters to **mqiptKeycmd**, as illustrated in the following examples:

- To create a PKCS#12 file:

```
mqiptKeycmd -keydb -create -db key.p12 -pw password -type pkcs12
```

- To create a self-signed personal certificate for testing purposes:

```
mqiptKeycmd -cert -create -db key.p12 -pw password -type pkcs12
            -label mqipt -dn "CN=Test Certificate,OU=Sales,O=Example,C=US"
            -sig_alg SHA256WithRSA -size 2048
```

The 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. When you create a certificate, take care to choose a public key encryption algorithm, key size, and digital signature algorithm that are appropriate for your organization's security needs. See “Digital certificate considerations for MQIPT” for more information.

This example uses a self-signed certificate that is suitable for test purposes. However, in a production environment you should use a Certificate Authority signed certificate instead.

Note that MQIPT v2.0 and older versions do not support SHA-2 digital signatures, so this certificate is not suitable for establishing secure socket connections to previous MQIPT releases; an older signature algorithm, such as SHA1WithRSA, would be required.

- To create a certificate request for a CA signed certificate for production purposes:

  ```
  mqiptKeycmd -certreq -create -db key.p12 -pw password -type pkcs12 -file cert.req
              -label mqipt -dn "CN=Test Certificate,OU=Sales,O=Example,C=US"
              -sig_alg SHA256WithRSA -size 2048
  ```

  The command creates a digital certificate request with a 2048-bit RSA public key and a digital signature that uses RSA with the SHA-256 hash algorithm. When you create a certificate, take care to choose a public key encryption algorithm, key size, and digital signature algorithm that are appropriate for your organization's security needs. See “Digital certificate considerations for MQIPT” for more information.

  Note that MQIPT 2.0 and older versions do not support SHA-2 digital signatures so this certificate is not suitable for establishing secure sockets connections to previous MQIPT releases: an older signature algorithm such as SHA1WithRSA would be required. The resulting certificate request is stored in a file named cert.req: this file should be sent to the CA to be signed.
- To receive the CA signed personal certificate file `cert.crt` into the key-ring file:

  ```
  mqiptKeycmd -cert -receive -db key.p12 -pw password -type pkcs12 -file cert.crt
  ```

  You must ensure that the CA certificate of the CA which signed the personal certificate is present in the CA key-ring file, for example:

  ```
  mqiptKeycmd -cert -add -db key.p12 -pw password -type pkcs12 -file ca.crt -label rootCA
  ```

  Many other examples of using the iKeyman CLI can be found in the IBM MQ documentation and also in Appendix B of the iKeyman documentation provided with MQIPT.

  For detailed information about using the iKeyman application, refer to the *iKeyman User's Guide* supplied with MQIPT. This is supplied as a PDF file in the `doc` directory of the MQIPT installation.
  – Wherever the command examples refer to the **ikeycmd** command, run **mqiptKeycmd** instead.
  – Wherever the command examples refer to the **ikeyman** command, run **mqiptKeyman** instead.

## Digital certificate considerations for MQIPT

### Certificate key size considerations for MQIPT

The public key size depends upon your organisation's security policy and depends on the encryption algorithm used. In general, larger key sizes are more secure. The following table lists the minimum key sizes that you should use:

| Algorithm | Minimum key size (bits) |
|---|---|
| Elliptic Curve | 256 |
| RSA | 2048 |

Specify the key size of your certificate when you create a certificate or certificate request.

- When using the **mqiptKeycmd** CLI command, the **-size** parameter specifies the key size.
- When using the mqiptKeyman GUI, the **Key Size** field in the Certificate Creation window specifies the key size.

## Selecting an appropriate certificate digital signature algorithm

To prevent forgery of digital certificates, it is important to use a strong digital signature algorithm. When you create or request a certificate, take care to select a good algorithm.

You should avoid using old digital signature algorithms based on MD5 or SHA-1 as these algorithms are no longer sufficiently secure for modern usage. If possible, use one of the newer SHA-2 based digital signature algorithms such as SHA-256 with RSA (SHA256WithRSA).

However, versions of MQIPT earlier than Version 2.1 do not support SHA-2 digital signatures, so for interoperability with previous MQIPT releases, use the SHA1WithRSA digital signature algorithm. However, you should plan to upgrade older versions of MQIPT and phase out use of MD5 and SHA-1 digital signatures.

- When using the **mqiptKeycmd** CLI command, the **-sig_alg** parameter specifies the digital signature algorithm.
- When using mqiptKeyman GUI, the **Signature Algorithm** field of the Certificate Creation window specifies the digital signature algorithm.

## Digital certificate and CipherSuite compatibility in MQIPT

Not all CipherSuites can be used with all digital certificates. There are various types of CipherSuite, grouped by their CipherSuite name prefix. Each type of CipherSuite imposes different restrictions on the type of digital certificate that can be used. These restrictions apply to all MQIPT SSL/TLS connections, but are particularly relevant to users of Elliptic Curve cryptography. When performing the secure socket handshake, MQIPT automatically selects a personal certificate to identify itself that is appropriate for the negotiated CipherSuite. In most cases MQIPT automatically interoperates with the remote peer. However, in certain scenarios you might need to use a specific MQIPT CipherSuite to interoperate with a remote IBM MQ system. The iKeyman application supplied with MQIPT is capable of creating certificates and certificate requests only with DSA and RSA public keys. Additionally, the IBM MQ **runmqakm** utility can create certificates and certificate requests with Elliptic Curve public keys. Consult your Certificate Authority for advice on creating other types of certificate.

The type of digital certificate to use depends upon the type of CipherSuite you are using:

- CipherSuites with names that begin SSL_ECDH_ECDSA_ and SSL_ECDHE_ECDSA_ require a digital certificate with an Elliptic Curve public key.

- CipherSuites with names that contain *anon* are anonymous; they do not require a digital certificate to identify the remote peer. Such CipherSuites can avoid the overheads of certificate lifecycle management in networks where an alternative means of authentication is used, but in general, avoid their use due to the lack of authentication.
- Other CipherSuites require a digital certificate with an RSA public key.

**Note:** The **mqiptKeyman** and **mqiptKeycmd** tools are unable to create certificates or certificate requests with an Elliptic Curve public key. You can use the **runmqakm** command provided with IBM MQ for this purpose, as documented in the IBM MQ product documentation.

## Certificate exit

The purpose of a certificate exit is to validate an SSL/TLS peer certificate that is received by MQIPT. You can configure an MQIPT route to act as an SSL/TLS client when it makes a new connection and to act as an SSL/TLS server when it receives a connection request. During the SSL/TLS handshaking process, an SSL/TLS client receives a peer certificate from the server, and the certificate can be used to authenticate the server. An SSL/TLS server can also receive a peer certificate from the client, and the certificate can be used to authenticate the client.

The certificate exit is called when MQIPT receives a peer certificate, allowing you to perform further validation. Any exceptions that are caught by the exit are caught by MQIPT and the connection request terminated. It is, therefore, good practice for the exit to catch all exceptions and to pass back an appropriate return code to MQIPT.

A sample is provided to show a certificate exit can be implemented for more information see “Scenario: Using a certificate exit to authenticate an SSL/TLS server” on page 111.

**Note:** MQIPT runs in a single JVM so a user-defined certificate exit might jeopardize the normal operation of MQIPT in one of these ways:
- Affect system resources
- Generate bottlenecks
- Degrade performance

You should test the effects of your certificate exit extensively before implementing it in a production environment.

### The com.ibm.mq.ipt.exit.CertificateExit class

An abstract class that must be implemented by the class that is defined with the SSLExitName property.

The class contains default implementations for running the exit and some public methods that you can optionally override, according to your requirements. The complete list of supported methods is as follows:

#### Methods

**public int init(IPTTrace)**

The init method is called by MQIPT when the exit is loaded by MQIPT and can be implemented to perform any initialization of the exit; for example, loading of data that is used during the validation process. The default implementation does nothing.

**public int refresh(IPTTrace)**

The refresh method is implemented to perform a refresh of any data; for example, reloading of any data for disk that is used during the validation process. This method is called when the MQIPT administrator has issued a refresh command. The default implementation does nothing.

**public void close(IPTTrace)**

The close method is implemented to perform any housekeeping when the route is about to be stopped or MQIPT is closing down. The default implementation does nothing.

**public CertificateExitResponse validate(IPTTrace)**

The validate method is called to perform validation of the peer certificate. The return object can be used to pass information back to MQIPT; for example, a return code and some text that can be added to the connection log. The default implementation returns a CertificateExitResponse with CertificateExitResponse.OK.

Supported methods for obtaining properties:

**public int getListenerPort()**
retrieves the route listener port - as defined by the ListenerPort property

**public String getDestination()**
retrieves the destination address - as defined by the Destination property

**public int getDestinationPort()**
retrieves the destination listener port address - as defined by the DestinationPort property

**public String getClientIPAddress()**
retrieves the IP address of the client making the connection request

**public int getClientPortAddress()**
retrieves the port address used by the client making the connection request

**public boolean isSSLClient()**
used to determine if the exit is being called as an SSL/TLS client or SSL/TLS server. If this returns true, the exit is on the client side of the connection, validating the certificate obtained from the server. If this returns false, the exit is on the server side of the connection, validating the certificate sent by the client. It is valid for a route to act as both an SSL/TLS server and an SSL/TLS client, decrypting and re-encrypting traffic. In this situation, although there is a single exit class, some instances of the class will be called as clients and some as servers. You can use isSSLClient to determine the situation for a given instance.

**public int getConnThreadID()**
used to retrieve the ID of the worker thread that is handling the connection request, which can be useful for debugging.

**public String getChannelName()**
retrieves the IBM MQ channel name that is used in the connection request. This is available only when the incoming request is not using SSL/TLS and MQIPT is acting as an SSL/TLS client.

**public String getQMName()**
: retrieves the name of the IBM MQ queue manager used in the connection request. This is available only when the client request is not using SSL/TLS and MQIPT is acting as an SSL/TLS client.

**public boolean getTimedout()**
: used by the exit to determine if the timeout has expired.

**public IPTCertificate getCertificate()**
: retrieves the SSL/TLS certificate that needs to be validated.

**public String getExitData()**
: retrieves the exit data, as defined by the SSLExitData property.

**public String getExitName()**
: retrieves the exit name, as defined by the SSLExitName property.

## The com.ibm.mq.ipt.exit.CertificateExitResponse class

This class is used to pass information back to MQIPT after a certificate has been validated.

### Constructors

**public CertificateExitResponse(*int rc*, string message)**

This constructor can be used to pass back a return code, and some message text. Possible reason codes are

- ExitRc.OK
- ExitRc.VALIDATE_ERROR
- ExitRc.VALIDATE_REJECTED

**public CertificateExitResponse(*int rc*)**

This constructor can be used to pass back a return code, with no message text. Possible reason codes are

- ExitRc.OK
- ExitRc.VALIDATE_ERROR
- ExitRc.VALIDATE_REJECTED

**public CertificateExitResponse()**

This constructor can be used to pass back return code ExitRc.OK, with no message text.

### Methods

**public String getVersion()**

This method returns the version of this class.

**public String toString**

This method will return a string representation of the response, for example, "Reason code: 4, Message: Failed CRL check.

## The com.ibm.mq.ipt.exit.IPTCertificate class

This class contains the SSL/TLS certificate to be validated.

### Methods

**public int getVersion()**

This method returns the version of this class.

**public byte [] getDerEncoding()**

This method returns the ASN.1/DER encoding of the X.509 certificate, or NULL if there is an error.

**public byte [] getPemEncoding()**

This method returns the PEM (BASE64) encoding of the X.509 certificate, or NULL if there is an error.

**public String getLabel()**

This method returns the certificate label, or NULL if there is an error.

**public String getName()**

This method returns the Distinguished Name of the certificate, or NULL if not available. For example:

```
CN=Test Queue Manager,OU=Sales,O=Example,L=London,C=GB
```

**public String getIssuerName()**

This method returns the issuer's Distinguished Name of the certificate, or NULL if not available. For example:

```
CN=Certificate Authority,OU=Security,O=Example,L=New York,C=US
```

**public IPTCertificate getSigner()**

This method returns the signer certificate, or NULL if not available. For a self-signed certificate it will return a reference to itself.

**public String toString()**

This method returns a string representation of the certificate.

## The com.ibm.mq.ipt.exit.IPTTrace class

### Methods

**public void entry(String *fid*)**

Where *fid* is used to identify where the call was made, for example the class and method name.

This method writes an entry to the trace output file with the appropriate level of indentation to record the point at which the flow of control enters a method. This call is optional, but if it is used, a matching call to "exit(String)" must also be used within the same method.

**public void exit(String *fid*)**

Where *fid* is used to identify where the call was made, for example the class and method name.

This method writes an exit to the trace output file with the appropriate level of indentation to record the point at which the flow of control leaves a method. This method is used only when a call to "entry(String)" has previously been used within the same method.

**public void exit(String *fid*, int *rc*)**

Where *fid* is used to identify where the call was made, for example the class and method name, and *rc* is the numeric return code from the method. This trace method should be used to record the exit from methods that return an integer.

This method writes an exit to the trace output file with the appropriate level of indentation to record the point at which the flow of control leaves a method and the numeric return code from that method. This method is used only when a call to "entry(String)" has previously been used within the same method.

**public void exit(String *fid*, boolean *rc*)**

Where *fid* is used to identify where the call was made, for example the class and method name, and *rc* is the Boolean return code from the method. This trace method should be used to record the exit from methods that return a Boolean.

This method writes an exit to the trace output file with the appropriate level of indentation to record the point at which the flow of control leaves a method and the Boolean return code from that method. This method is used only when a call to "entry(String)" has previously been used within the same method.

**public void data(String *fid*, String data)**

Where *fid* is used to identify where the call was made, for example the class and method name.

This method writes some string data to the trace output file.

**public void data(String *fid*, int data)**

Where *fid* is used to identify where the call was made, for example the class and method name.

This method writes some integer data to the trace output file.

**public void data(String *fid*, byte[])**

Where *fid* is used to identify where the call was made, for example the class and method name.

This method writes some binary data to the trace output file.

## Sample trace

To help diagnose problems in an exit, you can use the same tracing facility as MQIPT, alternatively you can implement your own tracing functions. If you decide to use the MQIPT trace functions there are entry and exit calls, that can be used on entry to and exit from a method. There are also various data calls to trace useful information as shown in the following example.

```
/**
  * This method is called to initialize the exit (for example, for
  * loading validation information) and place itself in a ready
  * state to validate connection requests.
  */
 public int init(IPTTrace t) {
   final String fid = "MyExit.init";

   // Trace entry into this method
   t.entry(fid);

   // Trace useful information
  t.data(fid, "Starting exit - MQIPT version " + getVersion());

  // Perform initialization and load any data
  t.data(fid, "Ready for work");

  // Trace exit from this method
  t.exit(fid);

  return ExitRc.OK;
 }
```

This method produces trace in the format shown in the following example:

```
16:36:48.625    14   5000-1s    ------{ ConnectionThread.setCertificateExit()
16:36:48.625    14   5000-1s     Creating instance of certificate exit
16:36:48.625    14   5000-1s     Calling init() of certificate exit
16:36:48.625    14   5000-1s    -------{ MyExit.init()
16:36:48.625    14   5000-1s     Starting exit - MQIPT version 2.1.0.0
16:36:48.625    14   5000-1s    Ready for work
16:36:48.625    14   5000-1s    -------} MyExit.init() rc=0
16:36:48.625    14   5000-1s    ------} ConnectionThread.setCertificateExit() rc=0
```

### Certificate exit return codes

The following return codes are recognized by MQIPT when calling a certificate exit in the following situations:

| Return code | Description | init | validate | refresh |
|---|---|---|---|---|
| ExitRc.OK | Request completed successfully. | yes | yes | yes |
| ExitRc.INIT_ERROR | Init request failed, route will be disabled. | yes | | |
| ExitRc.REFRESH_ERROR | Refresh request failed, route will be disabled. | | | yes |
| ExitRc.VALIDATE_ERROR | Validation process failed, connection request rejected. | | yes | |
| ExitRc.VALIDATE_REJECTED | Validation request rejected, connection request rejected. | | yes | |

## LDAP and CRLs

MQIPT supports use of a Lightweight Directory Access Protocol (LDAP) server to perform Certificate Revocation List (CRL) authentication on a digital certificate. LDAP support has been implemented in a similar way to that in IBM MQ, as the same LDAP server can be used for both IBM MQ and MQIPT.

During the SSL/TLS handshake, the communicating partners authenticate each other with digital certificates. Authentication can include a check that the certificate received can still be trusted. Certification Authorities (CAs) revoke certificates for various reasons, including the following:

- The owner has moved to a different organization.
- The private key is no longer secret.

CAs publish revoked personal certificates in a Certificate Revocation List (CRL). CA certificates that have been revoked are published in an Authority Revocation List (ARL). Note that subsequent references to CRLs also apply to ARLs.

For further information about the use of LDAP servers with IBM MQ and about the management of CRLs and ARLs, see the Security section in the IBM MQ product documentation.

MQIPT can support up to two LDAP servers on each route. The first LDAP server is treated as the main server with the second LDAP server kept as a backup. The second server is only used if the main server cannot be reached. The backup server should be a mirror image of the main server.

Access to information stored on an LDAP server can be protected with a user ID and password by using the LDAP user ID and password properties

When MQIPT loads a PKCS#12 token from a key-ring file, any CA certificates are checked for CRL validity. If the CA certificate has an attached CRL, it is checked to see if it has expired and, if so, a newer CRL is retrieved from the LDAP server. Any CRL retrieved is loaded into the current token and attached to its CA certificate. The updated token can be saved into the key-ring file. See the **LDAPSaveCRL** property in "Route properties" on page 125.

If there are no entries that match the given CA when a query is sent to the main LDAP server, it is assumed there are no CRLs for that CA and the backup server is not used. However, if the main LDAP server cannot be reached or does not return within a given time frame, the backup server is used. Any errors from the backup server cause the client connection to be ended. This action can be overridden by setting the **LDAPIgnoreErrors** property to `true`.

Any CRLs retrieved by MQIPT are kept in a cache and shared by all connections on that route. If a cached CRL has expired, the CRL is removed from the cache and a new one is retrieved from the LDAP server. If a new CRL is not available the connection is still refused.

A CRL retrieved from the LDAP server is also checked for expiry and a warning message is displayed (MQCPW001). The expired CRL is still loaded into the system and any connection requests referencing that CRL are refused. You should replace the expired CRL in the LDAP server with a current one.

The **LDAPCacheTimeout** property can be used to control how often the CRL cache is cleared. The default value is 1 day. Setting this value to `0` means the cache entries are not cleared until the route is restarted.

An expired CRL can be stored in a key-ring file or on an LDAP server. If a new CRL has not been issued, further connection requests are refused. You can ignore expired CRLs by enabling the **IgnoreExpiredCRLs** property.

**Note:** If you enable either the **LDAPIgnoreErrors** property or the **IgnoreExpiredCRLs** property, a revoked certificate might be used to make an SSL/TLS connection.

## Multi-valued certificate Distinguished Name OU properties

You can match multiple organizational unit (OU) values in certificate Distinguished Names.

The following route properties now support the matching of multiple OU values:

- **SSLClientDN_OU**
- **SSLClientSiteDN_OU**
- **SSLServerDN_OU**
- **SSLServerSiteDN_OU**

To match multiple OU values, use a comma as a separator in the route property value. For example:

```
SSLClientDN_OU=Sales, Europe
```

This matches certificates with both `OU=Sales` and `OU=Europe`. The OU values are matched in the same sequence as multiple OU values in IBM MQ SSLPEER filters.

Do not specify the same route property more than once in the `[route]` section. The correct way to match multiple OU values is to specify the property once, as shown in the preceding example. If you enter the same attribute more than once in the same `mqipt.conf` section, the last value takes effect. For example, the following entries would result in only matching `Europe` because the second line overrides the first:

```
SSLClientDN_OU=Sales
SSLClientDN_OU=Europe
```

If you must match a literal comma inside an OU value, insert a backslash (\) as an escape character immediately before the comma. For example:

```
SSLClientDN_OU=Sales\, Europe
```

This matches a single value: `OU=Sales, Europe`. A backslash that is not immediately followed by a comma matches a literal backslash.

If you are upgrading from a previous release of MQIPT and rely on the ability to match commas in OU values, you must insert backslash escape characters into the OU route properties in order to preserve the previous behavior.

## Network Dispatcher support

MQIPT can be used with IBM Network Dispatcher to provide enhanced availability and load balancing across many servers by the use of custom advisors.

This section assumes that you are familiar with Network Dispatcher and custom advisors.

Two custom advisors are supplied with MQIPT; they can be found in the `lib` subdirectory. Follow the instructions in the *Network Dispatcher User's Guide* for installing custom advisors. Figure 5 on page 48 shows an example of the use of the Network Dispatcher for monitoring port address 1414 for MQIPT. Note that each instance of MQIPT must have the same configuration file.

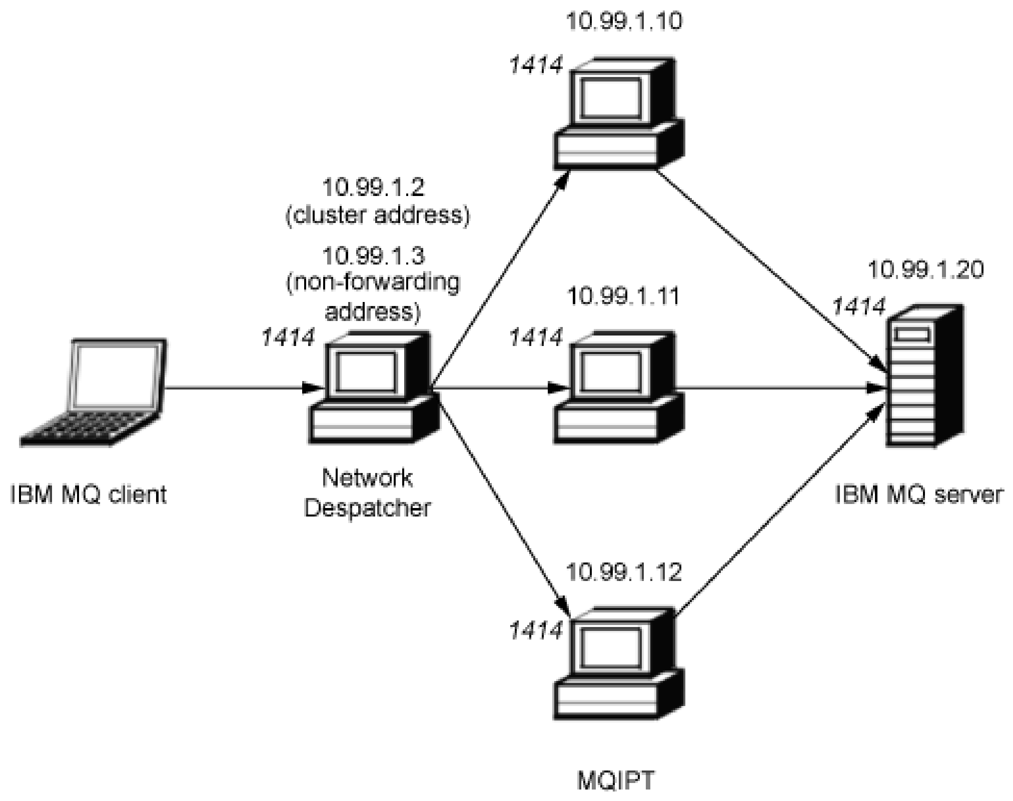


*Figure 5. Using the Network Dispatcher with MQIPT*

Follow the instructions in the *Network Dispatcher User's Guide* for configuring the dispatcher component to define port 1414 and the load-balanced server. You can use either the menu options of the Administration Client or the **ndcontrol** line mode command. For example:

```
ndcontrol port add 10.99.1.2 : 1414
ndcontrol server add 10.99.1.2 : 1414 : 10.99.1.10
ndcontrol server add 10.99.1.2 : 1414 : 10.99.1.11
ndcontrol server add 10.99.1.2 : 1414 : 10.99.1.12
```

The route definition in the MQIPT configuration file would look like this:

```
[route]
ListenerPort=1414
Destination=10.99.1.20
DestinationPort=1414
NDAdvisor=true
```

You can start (and stop) a custom advisor only from the command line. For example:

```
ndcontrol advisor start mqipt_normal 1414
```

This command starts the MQIPT advisor in normal mode, in which the base advisor performs its own timings to calculate the weighting factors of each MQIPT. To use the MQIPT advisor in replace mode, add the following line to the MQIPT route definition:

```
NDAdvisorReplaceMode=true
```

You must also start the `mqipt_replace` custom advisor instead of `mqipt_normal`. For example:

```
ndcontrol advisor start mqipt_replace 1414
```

When using an advisor to monitor an SSL/TLS listener port (that is, it has `SSLServer=true` in the `mqipt.conf` configuration file), you must place a "trigger" file in the working directory of the Network Dispatcher. This "trigger" file has a specific name, relating to the route being monitored. For example, if route 1414 has `SSLServer=true`, a file called `mqipt1414.ssl` must be placed in the `C:\windows\system32` directory (on Windows systems). See the `mqipt1414Sample.ssl` file for more information.

## Java Security Manager

The Java Security Manager can be used with any MQIPT feature to provide a further level of security.

MQIPT uses the default Java Security Manager as defined in the java.lang.SecurityManager class. The Java Security Manager feature in MQIPT can be enabled or disabled using the global property **`SecurityManager`**. See “Global properties” on page 124 for more information.

The Java Security Manager uses two default policy files:

- A global system policy file named *`$MQIPT_PATH`*`/java/jre/lib/security/java.policy` (where *`$MQIPT_PATH`* is the directory where MQIPT is installed) is used by all instances of a virtual machine on a host.
- A user-specific policy file called `.java.policy`, which can exist in the user's home directory.

An additional MQIPT policy file can also be used. You should use the MQIPT policy file instead of the default policy files described earlier. See **`SecurityManagerPolicy`** in “Global properties” on page 124 for more information.

The syntax of the policy file is quite complex and although it can be changed using a text editor, it is usually easier to use the Policy Tool utility provided with Java for making any changes. The Policy Tool utility can be found in the *`$MQIPT_PATH`*`/java/jre/bin` directory and is fully documented within the Java documentation.

A sample policy file (`mqiptSample.policy`) has been provided with MQIPT to show you what permissions must be set for running MQIPT.

You must edit the sample policy file to match your configuration. In particular, note that the MQIPT home directory (the location of mqipt.conf) might not be the same as the MQIPT installation directory, so take care to specify the correct directories when configuring FilePermission entries in the security policy.

You must change the following entries:

- The **`java.io.FilePermission`** entry which grants read and write access to the errors directory. The file path in this entry must refer to the MQIPT home directory, because this is where the errors directory is located. MQIPT creates FFST Failure Data Capture files (`AMQ*.FDC`) and trace files (`AMQ*.TRC*`) in the errors directory. You must ensure that MQIPT has permission to create trace and FFST files in the errors directory, so that troubleshooting is possible.

- The **java.io.FilePermission** entry which grants read and write access to the logs directory. The file path in this entry must refer to the MQIPT home directory, because this is where the logs directory is located. MQIPT creates connection log files (`mqipt*.log`) in the logs directory if the ConnectionLog global property is enabled.
- The **java.io.FilePermission** entries which grant read and execute access to any directories in the MQIPT installation directory, such as the `bin`, `exits`, `lib`, and `ssl` directories. The file paths in these entries must be changed to refer to the MQIPT installation directory. Some of these entries may be omitted if they are not required.
- The **java.net.SocketPermission** entries must be modified to control connections into each listening MQIPT route. The listen and accept permissions are required for the listener port and listener address for each MQIPT route.
- The **java.net.SocketPermission** entries must be modified to control connections out of each MQIPT route. The connect permission is required for any route destinations, proxy servers or LDAP servers that the MQIPT route connects to. The resolve permission is required when specifying addresses using a host name.

Depending on your configuration, you might also need to add the following entries:

- A **java.io.FilePermission** entry to grant read access to mqipt.conf, or the MQIPT home directory containing mqipt.conf. If you need to configure MQIPT remotely using the Administration Client then MQIPT will also need write access to mqipt.conf so that it can save configuration changes.
- A **java.io.FilePermission** entry to grant read access to the security policy file itself. This is useful if an MQIPT refresh causes the security policy file to be re-read.
- Some **java.io.FilePermission** entries to grant read access to any SSL/TLS keyring files and password stash files. This is only required when using a route which has the SSLClient or SSLServer properties enabled.
- Some **java.io.FilePermission** entries to grant read or execute access to any MQIPT exit classes. This is only required when an MQIPT exit is enabled. You might need to grant additional permissions if required by the exit.

**Note:** Windows **java.io.FilePermission** entries must use two backslash characters (\\) for every backslash in the path. This is because a single backslash is used as an escape character.

The sample file assumes that MQIPT has been installed on a Windows system in `C:\Program Files\IBM\MQ Internet Pass-Thru`. It also assumes that the MQIPT home directory (the location of the `mqipt.conf` file) is the same as the MQIPT installation directory.

If you have installed MQIPT in another location, you must change the directory in the codeBase definition to refer to your MQIPT installation directory. Take care to include the correct prefix (`file:/`) and the correct file suffix (`/lib/com.ibm.mq.ipt.jar`). On UNIX and Linux systems, a typical codeBase URL might be `file:/opt/mqipt/lib/com.ibm.mq.ipt.jar`, assuming that MQIPT is installed in `/opt/mqipt`.

Permissions are usually defined with three attributes. To control socket connections, their values are:

**class permission**
: `java.net.SocketPermission`

**name to control**
: This is made up with the format `hostname:port`, where each component of the name can be specified by a wildcard. The hostname can be a domain name or an IP address. The leftmost position of the host name can be specified by an asterisk (*). For example, `harry.company1.com` would be matched by each of these strings:

  - `harry`
  - `harry.company1.com`
  - `*.company1.com`
  - `*`
  - `198.51.100.123` (assuming this is the IP address of `harry.company1.com`)

  The port component of the name can be specified as a single port address or a range of port addresses, for example:

| | |
|---|---|
| **1414** | only port 1414 |
| **1414-** | all port addresses greater than or equal to 1414 |
| **-1414** | all port addresses less than or equal to 1414 |
| **1-1414** | all port addresses between 1 and 1414, inclusive |

**allowed action**
: The actions used by `java.net.SocketPermission` are:

  - accept, this allows permission to accept connections from the specified target
  - connect, this allows permission to connect to the specified target
  - listen, this allows permission to listen on the specified port or ports for connection requests
  - resolve, this allows permission to use the DNS name service to resolve domain names into IP addresses

Control of the Java Security Manager can also be made through the `java.security.manager` and `java.security.policy` Java system properties, but it is recommended you use the SecurityManager and SecurityManagerPolicy properties for controlling MQIPT.

To include diagnostic information in trace and FFST records, MQIPT must access certain MQIPT system properties and environment variables. You must always include the following properties in the Java security policy:

```
permission java.util.PropertyPermission "java.home", "read";
permission java.util.PropertyPermission "java.version", "read";
permission java.util.PropertyPermission "java.runtime.version", "read";
permission java.util.PropertyPermission "java.vm.info", "read";
permission java.util.PropertyPermission "java.vm.vendor", "read";
permission java.util.PropertyPermission "os.arch", "read";
permission java.util.PropertyPermission "os.name", "read";
permission java.util.PropertyPermission "os.version", "read";
permission java.lang.RuntimePermission "getenv.MQIPT_PATH";
```

If you do not include all of these properties, MQIPT will not operate correctly, and problem diagnosis will be impaired.

## Security exits

Use a security exit to control access to a target destination, as defined by the **Destination** route property. The security exit is called at the point when MQIPT receives a connection request from a client, but before it makes the connection to the target destination.

Based on the initial connection properties, the security exit decides whether the connection is allowed to complete.

When a route is started, the security exit is called in order to initialize and to make itself ready to process a connection request. The initialization process should be used to load any user data and prepare this data for quick and easy access, thereby minimizing the time taken to process a connection request.

Each route can have its own security exit.

- The **SecurityExit** property is used to enable/disable the user-defined security exit.
- The **SecurityExitName** property is used to define the class name of the user-defined security exit.
- The **SecurityExitPath** property is used to define the directory name containing the class file. If this property is not set, then it is assumed the class file will be found in the exits subdirectory. The **SecurityExitPath** can also define the name of a JAR file containing the user-defined security exit.
- The **SecurityExitTimeout** property is used by MQIPT to determine how long it should wait for a response from the security exit when validating a connection request.

See “Route properties” on page 125 for details of the security exit properties.

MQIPT uses the SecurityExit class to call a user-defined security exit. This class must be extended by the user-defined security exit and most of its methods overridden to provide the functionality required. A SecurityExitResponse object is used to pass back data to MQIPT and this data is used by MQIPT to decide if the connection request should be accepted or rejected. The SecurityExitResponse object can also contain a new destination and destination port address, used to override the route defined by the the security exit properties.

Three sample security exits are provided to show you how a security exit can be implemented.

- SampleSecurityExit shows how to control access to a IBM MQ queue manager, based on the name of the IBM MQ channel. It allows only a connection with a channel name starting with the string "MQIPT." See “Scenario: Using a security exit” on page 103 for more information.
- SampleRoutingExit allows dynamic routing of client connection requests to a pool of defined IBM MQ servers, each server hosting a queue manager of the same name and same attributes. The sample includes a configuration file that contains a list of server names. See “Scenario: Routing client connection requests to IBM MQ queue manager servers by using security exits” on page 105 for more information.
- SampleOneRouteExit allows dynamic routing to a IBM MQ queue manager that is derived from the IBM MQ channel name used in the connection request. The sample includes a configuration file that contains a map of queue manager names to server names. See “Scenario: Dynamically routing client connection requests” on page 108 for more information.

**Note:** MQIPT runs in a single JVM so a user-defined security exit might jeopardize the normal operation of MQIPT in one of these ways:

- Affect system resources
- Generate bottlenecks
- Degrade performance

You should test the effects of your security exit extensively before implementing it in a production environment.

## The com.ibm.mq.ipt.exit.SecurityExit class

This class and its public methods must be extended by the user-defined security exit to get access to some common data and allow some MQIPT initialization to take place. Before each method is called by MQIPT, some properties will be made available for the method to use. Their values can be retrieved by using the appropriate get methods defined in this class.

### Methods

**public int init(IPTTrace)**

The following properties are available:

- listener port
- destination
- destination port
- version

The init method will be called by MQIPT when a route is started. On return from this method the security exit must be ready to validate a connection request. Valid possible return codes are ExitRc.OK or ExitRc.INIT_ERROR.

**public int refresh(IPTTrace)**

The following properties are available:

- listener port
- destination
- destination port

The refresh method will be called by MQIPT when it has been asked to refresh itself by the MQIPT Administration Client. This action will usually be called when a property has been changed in the configuration file. MQIPT will load all the properties from the configuration file and determine which ones have been changed and whether a route needs to be restarted immediately, or whether it can wait until the next time MQIPT is restarted.

This method should perform a reload of any external data it uses (that is, data loaded during the init method). Valid possible return codes are ExitRc.OK or ExitRc.REFRESH_ERROR.

**public void close(IPTTrace)**

The following properties are available:

- listener port
- destination
- destination port

The close() method will be called by MQIPT when it has been asked to stop by the MQIPT Administration Client. It should free up any system resource it has acquired during its operation. MQIPT will wait for this method to complete before shutting down.

This method will also be called if a security exit was enabled, but has now been disabled in the configuration file.

**public SecurityExitResponse validate(IPTTrace)**
The following properties are available:

- listener port
- destination
- destination port
- timeout
- client IP address
- client port address
- channel name
- queue manager name

The validate method will be called by MQIPT when it receives a connection request to validate. The channel name and queue manager name will not be available if the SSLProxyMode property has been enabled, as this feature is only used to tunnel SSL/TLS data and therefore the data usually obtained from the initial data flow will be unreadable.

The security exit must return a SecurityExitResponse object, containing the following information:

- reason code (must be set)
- new destination address (optional)
- new destination listener port address (optional)
- message (optional)

The reason code will determine if the connection will be accepted or rejected by MQIPT. The newDestination and newDestinationPort fields can optionally be set to define a new target queue manager. If you do not set these properties, the route Destination and DestinationPort properties defined in the configuration file will be used. Any message will be appended to the connection log file entry.

Supported methods for obtaining properties:

**public int getListenerPort()**
retrieves the route listener port - as defined by the ListenerPort property

**public String getDestination()**
retrieves the destination address - as defined by the Destination property

**public int getDestinationPort()**
retrieves the destination listener port address - as defined by the DestinationPort property

**public String getClientIPAddress()**
retrieves the IP address of the client making the connection request

**public int getClientPortAddress()**
retrieves the port address used by the client making the connection request

**public int getTimeout()**
retrieves the timeout value. MQIPT will wait for the security exit to validate a request - as defined by the SecurityExitTimeout property

**public int getConnThreadID()**
retrieves the connection thread ID handling the connection request, which is useful for debugging purposes

**public String getChannelName()**
retrieves the IBM MQ channel name used in the connection request

**public String getQMName()**
retrieves the IBM MQ queue manager name used in the connection request

**public boolean getTimedout()**
can be used by the security exit to determine if the timeout has expired

## The com.ibm.mq.ipt.exit.SecurityExitResponse class

This class is used to pass a response back to MQIPT from a user-defined security exit and is used to determine if the connection request should be accepted or rejected. Objects of this type are only created in the validate method (see "The com.ibm.mq.ipt.exit.SecurityExit class" on page 53). There are convenience constructors for creating these objects and there are methods for each property. See the sample security exits for more information.

Creating a default SecurityExitResponse object rejects the connection request.

### Constructors

- **public SecurityExitResponse (String dest, int destPort, int rc, String msg)**
  where:
  - `dest` is the new target destination
  - `destPort` is the new destination port address
  - `rc` is the reason code
  - `msg` is a message that will be added to the connection log entry
- **public SecurityExitResponse (String dest, int destPort, int rc)**
- **public SecurityExitResponse (int rc, String msg)**
- **public SecurityExitResponse (int rc)**

### Methods

**public void setDestination(String dest)**
sets a new destination address for the connection request

**public void setDestinationPort(int port) throws IPTException**
sets a new destination listener port address for the connection request - throw an IPTException for an invalid port address

**public void setMessage(String msg)**
adds a message to the connection log record

**public void setReasonCode(int rc)**
sets the reason code for the connection request.

## Security exit return codes

The following return codes are recognized by MQIPT when calling a security exit in the following situations:

| Return code | Description | init | validate | refresh |
|---|---|---|---|---|
| ExitRc.OK | Request completed successfully. | yes | yes | yes |
| ExitRc.INIT_ERROR | Init request failed, route will be disabled. | yes | | |
| ExitRc.REFRESH_ERROR | Refresh request failed. | | | yes |
| ExitRc.NOT_AUTHORIZED | Validation process failed, connection request rejected. | | yes | |
| ExitRc.DISABLE_SSL | Validation request successful, connection to target will not use SSL or TLS. | | yes | |

## Tracing

To help diagnose problems in a user-defined security exit, you can enable a trace facility, similar to that used by MQIPT.

Enable tracing by setting the route **Trace** property to a value in the range 1 - 5. See the entry for **Trace** in "Route properties" on page 125.

There will probably be more than one instance of the security exit running at the same time so individual entries in the trace file can be identified by using the thread identifier.

The tracing functions are initialized by MQIPT when the security exit is started; all you have to do is choose what information you want to trace. There are many tracing examples in the sample user exits. See "Security exits" on page 52.

The minimum requirements for tracing are an `entry` call, an `exit` call, and the data that you want to trace. For example:

```
/**
  * This method is called to initialize the exit (for example, for
  * loading validation information) and place itself in a ready
  * state to validate connection requests.
  */
public int init(IPTTrace t) {
    final String strMethod = "CustomExit.init";

    // Trace entry into this method
    t.entry(strMethod);

    // Trace useful information
    t.data(strMethod, "Starting exit - MQIPT version " + getVersion());

    // Perform initialization and load any data
    t.data(strMethod, "Ready for work");

    // Trace exit from this method
    t.exit(strMethod);

    return 0;
}
```

## Port number control

When using MQIPT, it is possible to restrict the range of local port number that are used when making an outgoing connection.

Set the **OutgoingPort** property on the route to specify the initial local port number, and set **MaxConnectionThreads** to specify the number of ports to be used. For example, if you set **OutgoingPort** to `1600` and **MaxConnectionThreads** to `20`, then the range of local port numbers for that route is 1600 - 1619.

It is the responsibility of the MQIPT administrator to make sure that there are no conflicts of port numbers between routes.

If **OutgoingPort** is not defined, a default value of `0` means a system-allocated port number is used for each connection.

When using HTTP, the number of outgoing ports is twice as many as when not using HTTP. In the previous example, if the route used HTTP, the range of numbers would be 1600 - 1639.

See "Scenario: Allocating port numbers" on page 93 for more information.

### Multihomed systems

When using a multihomed system, you can specify which IP address an outgoing connection will bind to by using the **LocalAddress** property. Host names are not supported on this property.

## Other security considerations

MQIPT has several additional functions that help a designer build a secure solution.

- If there are many clients in an internal network all trying to make outgoing connections, they can all go through an MQIPT located inside the firewall. The firewall administrator then has to grant external access only to the MQIPT computer.
- MQIPT can connect only to queue managers for which it has been explicitly configured in its configuration file, unless MQIPT is acting as a SOCKS proxy or is using a security exit.
- MQIPT verifies that the messages it receives and transmits are valid, and conform to the IBM MQ protocol. This helps prevent MQIPT being used for security attacks outside the IBM MQ protocol. If MQIPT is acting as an SSL/TLS proxy, when all IBM MQ data and protocols have been encrypted, MQIPT can only guarantee the initial SSL/TLS handshake. In this situation, use the Java Security Manager.
- MQIPT allows channel exits to run their own end-to-end security protocols.
- You can restrict the total number of incoming connections by setting the `MaxConnectionThreads` property. This helps protect a vulnerable internal queue manager from denial of service attacks.

You must protect the MQIPT configuration file, `mqipt.conf`, from being read by unauthorized users because it might contain sensitive information, such as the **AccessPW** password that controls remote administrative access to MQIPT. Also, ensure that `mqipt.conf` is protected against unauthorized modification. Set the operating system file permissions for `mqipt.conf` such that only the user account that runs MQIPT can read or update the file.

If the MQIPT command port is enabled, you must also prevent unauthorized access to it. In particular, if the **RemoteShutdown** property is enabled then a remote user could shut down MQIPT. You should use a firewall to restrict the set of computers that can connect to the MQIPT command port. You must assess the risks of allowing remote MQIPT shutdown and consider disabling the **RemoteShutdown** property.

## Connection logs

MQIPT provides a connection log facility that contains lists of all successful and unsuccessful connection attempts. It is controlled by using the **ConnectionLog** and **MaxLogFileSize** properties. See "Global properties" on page 124 for more information.

Each time MQIPT is started, a new connection log is created. For identification, the filename includes the current time stamp, for example:

```
mqiptYYYYMMDDHHmmSS.log
```

where

`YYYY` is the year

`MM` is the month

`DD` is the day

`HH` is the hours

`mm` is the minutes

`SS` is the seconds

When a connection log reaches the maximum size as determined by the **MaxLogFileSize** property, a backup file, `mqipt001.log`, is created. A maximum of two backup files are maintained (`mqipt001.log` and `mqipt002.log`).

An entry in the connection log represents each part of a connection request. A connection request that is received by MQIPT and the resulting new connection that MQIPT makes to the destination address appears as two log entries, and subsequently two further entries when each connection is ended.

Here is the connection log for a successful connection request:

```
Wed May 15 13:13:51 BST 2013 conn accept 127.0.0.1(3842) 127.0.0.1(5000) OK 5000-0
Wed May 15 13:13:51 BST 2013 conn conn   127.0.0.1(3843) localhost(3500) OK 5000-0
Wed May 15 13:13:52 BST 2013 conn close  127.0.0.1(3842) 127.0.0.1(5000) OK 5000-0
Wed May 15 13:13:52 BST 2013 conn close  127.0.0.1(3843) localhost(3500) OK 5000-0
```

Here is a connection log for a failed connection request:

```
Wed May 15 14:56:40 BST 2013 conn accept 127.0.0.1(4138) 127.0.0.1(7000) OK    7000-0
Wed May 15 14:56:40 BST 2013 conn close  127.0.0.1(4138) 127.0.0.1(7000) ERROR 7000-0
                       Unrecognized SSL handshake request '54'
```

# Chapter 5. Installing, uninstalling, and migrating MQIPT

You can install, uninstall, or migrate MQIPT by completing one of the following tasks:

- "Installing MQIPT"
- "Uninstalling MQIPT version 2.0" on page 61
- "Uninstalling MQIPT version 2.1" on page 62
- "Migrating from MQIPT Version 2.0 to Version 2.1" on page 62
- "Upgrading your MQIPT Version 2.1 installation" on page 66

## Installing MQIPT

From version 2.1 of MQIPT, you can install the product wherever you want on your computer, and can have several installations at the same time.

### About this task

Each installation can be used and maintained separately, so for example you can have different fix pack levels of MQIPT installed in different locations if you choose.

The installation location is not fixed: MQIPT version 2.1 can be installed anywhere on the system. It is no longer necessary to set the system **PATH** or **CLASSPATH** environment variables to refer to MQIPT.

The MQIPT commands can be invoked from any location and MQIPT automatically detects its own location. You might choose to add the MQIPT `bin` directory to the **PATH** environment variable for convenience, but it is not mandatory to do so.

You can also install MQIPT version 2.1 alongside version 2.0. You can have only one installation of MQIPT version 2.0 on the same system because of the installation method used by version 2.0.

If you run MQIPT as a system service, you can install only one such service on each system. You cannot install more than one MQIPT service on the same system, either from the same MQIPT installation or from different installations. Also, only the installation of MQIPT that installed the service can be used to remove it. For example, if you have two MQIPT installations, one in `C:\MQIPT1` and one in `C:\mqipt2`, and you run the command `C:\MQIPT1\bin\mqiptService -install C:\mqipt1`, then only the **mqiptService** command from the `C:\MQIPT1` installation can subsequently be used to remove the service. Attempting to remove the service using a different installation causes error MQCPE083.

### Procedure

To install MQIPT version 2.1, complete the following steps:

1. Create a new directory where you want MQIPT to be installed.

   For example, on a UNIX platform, you might use the following command:

   ```
   mkdir /opt/mqipt
   ```

2. Unpack the installation archive file into the MQIPT directory by using an appropriate tool for your platform.

   Note that the **tar** command on UNIX and Linux systems must be run as the root user. Choose the correct installation archive file for your platform. The readme file supplied with MQIPT specifies the exact operating system versions that are supported.

| Platform | Archive file |
|---|---|
| AIX® | `ms81_rios_aix_4.tar` |
| HP-UX | `ms81_ia64_hpux_11.tar` |
| Linux x86 (32 bit) | `ms81_x86_linux_2.tar` |
| Linux x86 (64 bit) | `ms81_amd64_linux_2.tar` |
| Linux zSeries | `ms81_s390x_linux_2.tar` |
| Solaris SPARC | `ms81_sparc_solaris_2.tar` |
| Solaris x86 (64 bit) | `ms81_amd64_solaris_2.tar` |
| Windows (32 bit) | `ms81_x86_nt_4.zip` |

   For example, on an AIX platform, you might use the following commands, if the archive file was downloaded to the `/tmp` directory:

```
cd /opt/mqipt
su root
tar xvf /tmp/ms81_rios_aix_4.tar
```

3. To increase security, set the file permissions for your installed files so that they are read-only.

   On Linux or UNIX systems, you can use the **chmod** command. For example:

```
chmod -R /opt/mqipt a-w
```

   On Windows platforms, right-click the installation directory and select **Properties**. You can change the file permissions on the **Security** tab.

4. If you subsequently receive error MQCPE080 `Unable to determine MQIPT installation directory`, set the **MQIPT_PATH** environment variable to the absolute path of the MQIPT installation directory.

   You do not normally have to set **PATH** or **CLASSPATH** variables for MQIPT because the installation includes a Java Runtime Environment (JRE). However, under some circumstances (for example, if you use symbolic links), MQIPT commands are unable to determine the installation directory. This can be corrected by setting the **MQIPT_PATH** environment variable.

   For example, if your installation directory is `/opt/mqipt`, you might use the following commands:

```
MQIPT_PATH=/opt/mqipt
export MQIPT_PATH
```

5. On Windows platforms, create MQIPT icons on the Start menu. Run the following command from a command line:

```
C:\mqipt_path\bin\mqiptIcons -install installation_name
```

   where
   *mqipt_path* is the directory where MQIPT is installed.
   *installation_name* is a name that you choose to distinguish this installation from any other

## Uninstalling MQIPT version 2.0

The procedure for uninstalling MQIPT depends on the platform on which it was installed, and the way it was installed.

**Note:** If you run MQIPT as a system service, only the installation of MQIPT that installed the service can be used to remove it. For example, if you have two MQIPT installations, one in `C:\MQIPT1` and one in `C:\mqipt2`, and you run the command `C:\MQIPT1\bin\mqiptService -install C:\mqipt1`, then only the **`mqiptService`** command from the `C:\MQIPT1` installation can subsequently be used to remove the service. Attempting to remove the service using a different installation causes error MQCPE083.

All of the following sections, except the last, assume that you installed MQIPT by using a system installable image.

### Uninstalling from AIX systems

1. Make appropriate backups in case you later have to restore any data. See "Making backups" on page 142 for details.
2. Make sure that you are logged in as `root`.
3. Prevent the system from trying to start MQIPT automatically by removing the entry for MQIPT from the `inittab` file:
   ```
   cd /usr/opt/mqipt/bin
   ./mqiptService -remove
   ```
4. Run the **`installp`** command:
   ```
   installp -u mqipt-RT
   ```

### Uninstalling from HP-UX systems

1. Make appropriate backups in case you later have to restore any data. See "Making backups" on page 142 for details.
2. Make sure that you are logged in as `root`.
3. Prevent the system from trying to start MQIPT automatically:
   ```
   cd /opt/mqipt/bin
   ./mqiptService -remove
   ```
4. Run the **`swremove`** command:
   ```
   swremove MQIPT
   ```

### Uninstalling from Linux systems

1. Make appropriate backups in case you later have to restore any data. See "Making backups" on page 142 for details.
2. Make sure that you are logged in as `root`.
3. Prevent the system from trying to start MQIPT automatically:
   ```
   cd /opt/mqipt/bin
   ./mqiptService -remove
   ```
4. Run the **`rpm`** command:
   ```
   rpm -e WebSphereMQ-IPT-2.0.0-1
   ```

### Uninstalling on Solaris systems

1. Make appropriate backups in case you later have to restore any data. See "Making backups" on page 142 for details.
2. Make sure that you are logged in as `root`.
3. Prevent the system from trying to start MQIPT automatically:

```
cd /opt/mqipt/bin
./mqiptService -remove
```

4. Run the **pkgrm** command:
   ```
   pkgrm mqipt
   ```

### Uninstalling from Windows systems

1. Make appropriate backups in case you later have to restore any data. See "Making backups" on page 142 for details.
2. Prevent the system from trying to start MQIPT as a service:
   a. Stop MQIPT from the Windows services panel.
   b. Open an administration command prompt, go to the MQIPT `bin` subdirectory, and enter:
      ```
      mqiptService -remove
      ```
3. Run the uninstallation process from the Windows **Start** menu.

### Uninstalling following a generic UNIX installation

Make appropriate backups in case you later have to restore any data. See "Making backups" on page 142 for details.

If MQIPT was not installed using a system installable image, you can uninstall it by deleting the directory structure into which it was installed.

If MQIPT was configured to run as a system service, remove the service before uninstalling the code.

## Uninstalling MQIPT version 2.1

### Procedure

To uninstall MQIPT version 2.1, complete the following steps:

1. Make appropriate backups in case you later have to restore any data. See "Making backups" on page 142 for details.
2. Prevent the system from trying to start MQIPT automatically. You can see examples of appropriate commands for various platforms in "Uninstalling MQIPT version 2.0" on page 61
3. On Windows platforms, remove the MQIPT icons from the **Start** menu by clicking the MQIPT icon, **Remove these icons** on the **Start** menu.
4. Delete the directory where MQIPT is currently installed.

## Migrating from MQIPT Version 2.0 to Version 2.1

You can migrate your MQIPT Version 2.0 installation to Version 2.1.

To perform the migration, complete the following steps:

1. Make appropriate backups in case you later have to restore any data. See "Making backups" on page 142 for details.
2. Stop MQIPT by running the command:
   ```
   mqiptAdmin -stop
   ```
3. If you have installed MQIPT as a service, you must remove it before uninstalling MQIPT. Note that only the installation of MQIPT that installed the service can be used to remove it. For example, if you have two MQIPT

installations, one in `C:\MQIPT1` and one in `C:\mqipt2`, and you run the command `C:\MQIPT1\bin\mqiptService -install C:\mqipt1`, then only the **`mqiptService`** command from the `C:\MQIPT1` installation can subsequently be used to remove the service. Attempting to remove the service using a different installation causes error MQCPE083.

```
mqiptService -remove
```

4. Run the uninstallation program for MQIPT Version 2.0.
5. After you have installed MQIPT Version 2.1, copy the saved configuration files back to their original locations.
6. You are advised to use the MQIPT Administration Client to manage changes to MQIPT. The configuration file from version 2.0 is compatible with the GUI.

Some implementations require a local MQIPT service under the control of your own organization and a remote MQIPT service which could be under the control of your client organization. In this situation, it is very difficult to migrate both MQIPT services at the same time but this is not a problem for MQIPT. Unless otherwise stated, older versions of MQIPT are compatible with the latest version. This makes the MQIPT migration process much easier.

## CipherSuites

The following CipherSuites are no longer supported:
- SSL_DH_anon_WITH_RC4_40_MD5
- SSL_DHE_DSS_EXPORT1024_WITH_DES_CBC_SHA
- SSL_DHE_DSS_EXPORT1024_WITH_RC4_56_SHA
- SSL_RSA_EXPORT_WITH_RC2_CBC_40_MD5
- SSL_RSA_EXPORT1024_WITH_DES_CBC_SHA
- SSL_RSA_EXPORT1024_WITH_RC4_56_SHA

If you include one of these CipherSuites in the **`SSLClientCipherSuites`** or **`SSLServerCipherSuites`** properties, the route will fail to start with error MQCPE076. No IBM MQ CipherSpecs correspond to these CipherSuites, so this only affects interoperability between a pair of MQIPT instances that use SSL/TLS.

The following CipherSuites are not enabled by default in JSSE, whereas they were enabled by default in SSLite. This is because these CipherSuites are no longer considered sufficiently secure and you should avoid using them. This change affects SSLClient routes without the **`SSLClientCipherSuites`** property and SSLServer routes without the **`SSLServerCipherSuites`** property, as these routes rely on the CipherSuites that are enabled by default. If you require any of these CipherSuites, you must specify the **`SSLClientCipherSuites`** or **`SSLServerCipherSuites`** route properties and include them in the property value.
- SSL_DH_anon_EXPORT_WITH_DES40_CBC_SHA
- SSL_DH_anon_EXPORT_WITH_RC4_40_MD5
- SSL_DH_anon_WITH_3DES_EDE_CBC_SHA
- SSL_DH_anon_WITH_AES_128_CBC_SHA
- SSL_DH_anon_WITH_AES_256_CBC_SHA
- SSL_DH_anon_WITH_DES_CBC_SHA
- SSL_DH_anon_WITH_RC4_128_MD5
- SSL_DHE_DSS_EXPORT_WITH_DES40_CBC_SHA
- SSL_DHE_DSS_WITH_DES_CBC_SHA
- SSL_DHE_DSS_WITH_RC4_128_SHA

- SSL_DHE_RSA_EXPORT_WITH_DES40_CBC_SHA
- SSL_DHE_RSA_WITH_DES_CBC_SHA
- SSL_RSA_EXPORT_WITH_DES40_CBC_SHA
- SSL_RSA_EXPORT_WITH_RC4_40_MD5 (Note 1)
- SSL_RSA_FIPS_WITH_3DES_EDE_CBC_SHA (Note 1,2)
- SSL_RSA_FIPS_WITH_DES_CBC_SHA (Note 1,2)
- SSL_RSA_WITH_DES_CBC_SHA (Note 1)
- SSL_RSA_WITH_NULL_MD5 (Note 1)
- SSL_RSA_WITH_NULL_SHA (Note 1)

**Notes:**

1. These CipherSpecs correspond to IBM MQ CipherSpecs as described in "SSL/TLS support" on page 24.
2. These names are historical; they are no longer FIPS-compliant and you should avoid using them.

If you require the same IBM MQ CipherSpecs enabled in MQIPT 2.1 as the previous MQIPT release, set the following properties in the `[global]` section of `mqipt.conf`. Enter each of these parameters on a single line in `mqipt.conf`, separating each CipherSuite value with a space:

```
SSLServerCipherSuites=SSL_RSA_WITH_DES_CBC_SHA SSL_RSA_FIPS_WITH_3DES_EDE_CBC_SHA
SSL_RSA_FIPS_WITH_DES_CBC_SHA SSL_RSA_WITH_NULL_MD5 SSL_RSA_WITH_NULL_SHA
SSL_RSA_EXPORT_WITH_RC4_40_MD5 SSL_RSA_WITH_RC4_128_MD5 SSL_RSA_WITH_RC4_128_SHA
SSL_RSA_WITH_3DES_EDE_CBC_SHA SSL_RSA_WITH_AES_128_CBC_SHA SSL_RSA_WITH_AES_256_CBC_SHA
SSL_RSA_WITH_DES_CBC_SHA SSL_RSA_WITH_RC4_128_SHA SSL_RSA_WITH_3DES_EDE_CBC_SHA
SSLClientCipherSuites=SSL_RSA_WITH_DES_CBC_SHA SSL_RSA_FIPS_WITH_3DES_EDE_CBC_SHA
SSL_RSA_FIPS_WITH_DES_CBC_SHA SSL_RSA_WITH_NULL_MD5 SSL_RSA_WITH_NULL_SHA
SSL_RSA_EXPORT_WITH_RC4_40_MD5 SSL_RSA_WITH_RC4_128_MD5 SSL_RSA_WITH_RC4_128_SHA
SSL_RSA_WITH_3DES_EDE_CBC_SHA SSL_RSA_WITH_AES_128_CBC_SHA SSL_RSA_WITH_AES_256_CBC_SHA
SSL_RSA_WITH_DES_CBC_SHA SSL_RSA_WITH_RC4_128_SHA SSL_RSA_WITH_3DES_EDE_CBC_SHA
```

Adding these properties to the `[global]` section ensures that all SSL/TLS routes that do not explicitly define a list of enabled CipherSuites inherit these settings. The presence of these properties only affects routes with **`SSLServer`** or **`SSLClient`** set to `true`; other routes are unaffected.

With this set of CipherSuites, you can use all of the IBM MQ CipherSpecs supported by MQIPT 2.0 after migration to version 2.1.

## Changes to organizational unit (OU) matching

The following `[route]` properties now support the matching of multiple OU values by using commas as separators:

- **`SSLClientDN_OU`**
- **`SSLClientSiteDN_OU`**
- **`SSLServerDN_OU`**
- **`SSLServerSiteDN_OU`**

If you previously used any of these OU `[route]` properties and included commas, you must insert backslash (\) escape characters before any commas to preserve the previous behavior and avoid them being interpretted as separators. See "Multi-valued certificate Distinguished Name OU properties" on page 46 for more information.

## Changes to SecurityManager security policy

MQIPT Version 2.1 includes additional problem diagnosis information in its trace and FFST records. If you use a Java SecurityManager security policy, you must always include the following properties in the policy:

```
permission java.util.PropertyPermission "java.home", "read";
permission java.util.PropertyPermission "java.version", "read";
permission java.util.PropertyPermission "java.runtime.version", "read";
permission java.util.PropertyPermission "java.vm.info", "read";
permission java.util.PropertyPermission "java.vm.vendor", "read";
permission java.util.PropertyPermission "os.arch", "read";
permission java.util.PropertyPermission "os.name", "read";
permission java.util.PropertyPermission "os.version", "read";
permission java.lang.RuntimePermission "getenv.MQIPT_PATH";
permission java.lang.RuntimePermission "getStackTrace";
permission java.security.SecurityPermission "getPolicy";
```

If you do not include all of these properties, MQIPT will not operate correctly, and problem diagnosis will be impaired.

When you migrate from a previous version, ensure that all of these properties are included in any security policy file identified by the **SecurityManagerPolicy** property in the [global] section of the mqipt.conf file. If do you not have a **SecurityManagerPolicy** property, it is not necessary to create one.

## IBM MQ AMQ9616 or AMQ9631 errors with SSL/TLS routes

MQIPT Version 2.1 supports more TLS secure socket protocol versions than the previous releases. As a result, it is possible that existing IBM MQ channels might fail to connect through MQIPT after MQIPT has been upgraded to version 2.1.

This issue is most likely to affect SSLClient routes, because the listener on the destination queue manager has all protocol versions enabled so that it can perform handshakes for multiple channels with different CipherSpecs. The MQIPT Version 2.1 default is also to enable all protocol versions for security reasons. The secure sockets protocols are designed to negotiate the highest enabled protocol version supported by both sides. If you are using older SSL 3.0 and TLS 1.0 CipherSpecs with IBM MQ Version 7.1 or later, this might result in TLS 1.2 being used instead of the expected protocol. This protocol version mismatch causes the channel to report an error.

If IBM MQ channels that worked with the previous MQIPT release fail to start and produce AMQ9616 or AMQ9631 errors after migration, set the protocol versions that are to be used: set the **SSLClientProtocols** route property to restrict the protocol versions for an SSLClient route and the **SSLServerProtocols** route property to restrict the protocol versions for an SSLServer route. In most cases, setting SSLClientProtocols=SSLv3 will resolve the problem. If you have MQIPT V2.0 and an interim fix for APAR IV25345 installed, you might need to add TLSv1 to the **SSLClientProtocols** list.

To assist with migration, add the following settings to the [global] section of mqipt.conf so that they are inherited by all SSL/TLS routes:

```
SSLClientProtocols=SSLv3,TLSv1
SSLServerProtocols=SSLv3,TLSv1
```

These settings only affect routes with **SSLServer** or **SSLClient** set to `true`; other routes are unaffected.

Refer to the table of CipherSpecs and CipherSuites in "SSL/TLS support" on page 24 for more information about the protocol versions used by IBM MQ CipherSpecs and how best to configure the protocol versions.

### Connection errors when using 512-bit RSA keys after migration

If you have a digital certificate with a 512-bit RSA public key, you might see one of the following connection errors in the MQIPT connection log, after you migrate to version 2.1:

- `SSLException: Error generating ECDH server key exchange`
- `Unsupported SignatureAndHashAlgorithm in ServerKeyExchange message`

The failure arises because MQIPT V2.1 supports TLS 1.2 CipherSuites, which are enabled by default unless you have specifically set the **SSLClientCipherSuites** or **SSLServerCipherSuites** route properties. The IBM JSSE implementation of the TLS 1.2 CipherSuites does not support 512-bit RSA keys because 512 bits is an insecure key size; it provides only a weak strength of encryption and should be avoided.

Larger RSA key sizes such as 768 bits and 1024 bits are not affected and are supported, although for security reasons, it is preferable to use a minimum RSA key size of 2048 bits.

To check the key size for your existing certificates, you can use the supplied **mqiptKeycmd** tool:

1. Run the certificate list command to obtain a list of the certificate labels in the key-ring file. For example:
   ```
   mqiptKeycmd -cert -list -db key.p12 -pw password
   ```
2. For each certificate label (for example, `mqipt`), display the certificate details:
   ```
   mqiptKeycmd -cert -details -db key.p12 -pw password -label mqipt
   ```

   The **Key Size** field displays the certificate public key size. If the size shown is 512 then this issue is likely to occur.

The preferred solution is to replace the old RSA digital certificate with a new certificate using an RSA key size of at least 2048 bits. This offers a much better strength of encryption and also ensures compatibility with the TLS 1.2 protocol.

However, if necessary, you can work around the error by configuring the set of enabled protocol versions to exclude TLS 1.2, for example:

```
SSLClientProtocols=SSLv3,TLSv1,TLSv1.1
SSLServerProtocols=SSLv3,TLSv1,TLSv1.1
```

## Upgrading your MQIPT Version 2.1 installation

Shortdesc

### Procedure

To apply fix pack maintenance to your MQIPT Version 2.1 installation, complete the following steps:

1. Make backups of your data. See "Making backups" on page 142 for details.

2. Uninstall the old version of MQIPT. See “Uninstalling MQIPT version 2.1” on page 62 for details.
3. Install the new version of MQIPT. See “Installing MQIPT” on page 59 for details.
4. Copy your backed-up data files to their original locations, overwriting any newly installed copies of these files.

# Chapter 6. Scenarios: Getting started with MQIPT

The scenarios in this section show you how to set up some simple IBM MQ Internet Pass-Thru configurations. You can also use these tasks to confirm that the product has been installed successfully.

## Before you begin

Before you start to use the scenarios in this section, 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 a 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.)
- 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.

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 a user ID and password for client connections, you will need to set the MQSAMP_USER_ID environment variable to the user ID to be used for connection authentication before running the **`amqsputc`** and **`amqsgetc`** commands.

After you have done this, you can 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.

Edit the `mqipt.conf` file as follows:

- Copy `mqiptSample.conf`, which you can find in the MQIPT installation directory, to `mqipt.conf` in your chosen MQIPT home directory. The following scenarios use `C:\mqiptHome` as the MQIPT home directory.

- 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.
- Delete all routes from the `mqipt.conf` file.
- In the remaining `[global]` section, check that the following entries exist, adding them if necessary, and set them to the following values:
  - **`ClientAccess`** is set to `true`.
  - **`Destination`** is set to the network address of your queue manager. You can specify either a host name or an IP address.
  - **`DestinationPort`** is set to the port number used by your queue manager.

### The following scenarios are described in this section:

- "Scenario: Verifying that MQIPT is working correctly"
- "Scenario: Creating a key-ring file" on page 72
- "Scenario: Creating test certificates" on page 75
- "Scenario: Authenticating an SSL/TLS server" on page 76
- "Scenario: Authenticating an SSL/TLS client" on page 79
- "Scenario: Configuring an HTTP proxy" on page 83
- "Scenario: Configuring access control" on page 85
- "Scenario: Configuring a SOCKS proxy" on page 87
- "Scenario: Configuring a SOCKS client" on page 88
- "Scenario: Configuring MQIPT clustering support" on page 90
- "Scenario: Allocating port numbers" on page 93
- "Scenario: retrieving CRLs by using an LDAP server" on page 94
- "Scenario: running MQIPT in SSL/TLS proxy mode" on page 97
- "Scenario: running MQIPT in SSL/TLS proxy mode with a security manager" on page 99
- "Scenario: Apache rewrite" on page 101
- "Scenario: Using a security exit" on page 103
- "Scenario: Routing client connection requests to IBM MQ queue manager servers by using security exits" on page 105
- "Scenario: Dynamically routing client connection requests" on page 108
- "Scenario: Using a certificate exit to authenticate an SSL/TLS server" on page 111

## Scenario: Verifying that MQIPT is working correctly

Use this simple configuration setup to ensure that MQIPT was installed correctly.

### Before you begin

- Before you start to use this scenario, make sure that you have completed the prerequisite tasks listed in Chapter 6, "Scenarios: Getting started with MQIPT," on page 69.

## About this task

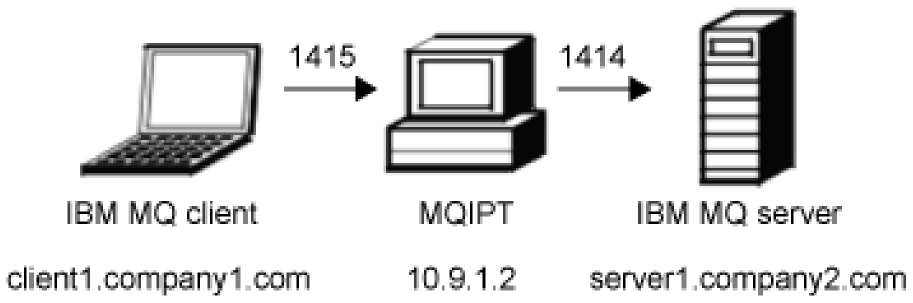


*Figure 6. Installation verification test network diagram*

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. Set up MQIPT. 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
   ```
   where `C\mqiptHome` indicates the location of the MQIPT configuration file, `mqipt.conf`.

   The following message indicates successful completion:
   ```
   5639-L92 (C) Copyright IBM Corp. 2000, 2017 All Rights Reserved
   MQCPI001 IBM MQ Internet
   Pass-Thru Version 2.1.0.3 starting
   MQCPI004 Reading configuration information from C:\mqiptHome\mqipt.conf
   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 protocols
   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 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.

## Scenario: Creating a key-ring file

In this scenario, you can request a certificate and create a key-ring file.

### Before you begin

- Before you start to use this scenario, make sure that you have completed the prerequisite tasks listed in Chapter 6, "Scenarios: Getting started with MQIPT," on page 69.

This task assumes you request a new certificate from a trusted Certificate Authority (CA) by using iKeyman, and that your personal certificate is returned to you in a file (for example, `server.cer`). This is sufficient to perform server authentication. If you require client authentication you must request a second certificate (for example, `client.cer`) and perform the following steps twice, to create two key-ring files.

### About this task

You can use either the iKeyman command-line interface (CLI) or the iKeyman GUI to request the certificate.

### Procedure

Use one of the following methods to create a key-ring file:

- **Use the iKeyman command-line interface (CLI)**
  1. Create a new PKCS#12 file:
     ```
     mqiptKeycmd -keydb -create -db server_name.pfx -pw key_password -type key_type
     ```
     where:
     - **-db** specifies the file (*server_name*.pfx) that corresponds to the key-ring property of the MQIPT route. For example, use `SSLClientKeyRing` for an SSLClient route and `SSLServerKeyRing` for an SSLServer route.
     - **-pw** specifies the password (*key_password*) that you must use later with the **mqiptPW** utility to store the encrypted key-ring password.
     - **-type** specifies the format of the key database; for example, `pkcs12`.
  2. Generate a new certificate request:
     ```
     mqiptKeycmd -certreq -create -db server_name.pfx -pw key_password -type key_type
                 -file cert_file_name.req -label label -dn DN_identity
                 -sig_alg signature_algorithm -size key_size
     ```
     where:
     - **-type** specifies the format of the key database; for example, `pkcs12`.
     - **-file** specifies a file name for the requested certificate.
     - **-label** specifies a unique name of your choice; it is preferable not to include space characters.
     - **-dn** specifies the appropriate Distinguished Name identity for the MQIPT route; for example, `"CN=Test Certificate,OU=Sales,O=Example,C=US"`.

  - **-sig_alg** specifies the hash algorithm; for example, SHA256WithRSA.
  - **-size** specifies the size of the public key; for example, 2048.

  If you use the example values given, 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.

  When creating a certificate, take care to choose an appropriate public key encryption algorithm, key size, and digital signature algorithm for your organization's security needs. See "Digital certificate considerations for MQIPT" on page 38 for more information.

  Send the certificate request file (*cert_file_name*.req) created by the command to your CA to be signed.

3. When you receive the signed personal certificate from the CA, add it into the server key ring:

```
mqiptKeycmd -cert -receive -db server_name.pfx -pw key_password
          -type key_type -file cert_file_name.crt
```

- **Use the iKeyman GUI**
  1. Open the iKeyman GUI by running the following command:

     ```
     mqiptKeyman
     ```
  2. Click **Key database file** > **New**.
  3. Select the type of the key database; for example, PKCS12.
  4. Enter the file name and location for the new key-ring file. This must correspond to the key-ring property of the MQIPT route; for example, use SSLClientKeyRing for an SSLClient route and SSLServerKeyRing for an SSLServer route. Click **OK**.
  5. Enter, and confirm, a password for the new key-ring file. This is the password that you must use later with the **mqiptPW** utility to store the encrypted key-ring password. Click **OK** to create the new personal-certificate key-ring file.
  6. Create the certificate request by clicking **Create** > **New Certificate Request**.
  7. Enter a label for the new certificate in the **Key Label** field. The label can be any unique name you choose; it is preferable not to include space characters.
  8. Select the key size and digital signature algorithm as appropriate for your organization's security needs. See "Digital certificate considerations for MQIPT" on page 38 for more information.
  9. Enter the appropriate Distinguished Name identity for the MQIPT route in the optional DN fields.
  10. Enter the file name for the certificate request to create, and click **OK**. The certificate request is generated and saved with the name you specify. Send this file to your CA to be signed.
  11. When you receive the signed personal certificate from the CA, you must receive it in the key-ring file. In the "Key database content" panel select Personal Certificates from the drop-down list. Then click **Receive**.
  12. Enter the name of the file where the signed certificate is stored, then click **OK**.

## What to do next

You must also ensure that the CA certificate of the CA that signed the personal certificate is present in the CA key-ring file. Depending on your MQIPT configuration, the CA key-ring file might be a different file from the personal certificate key-ring file. Check the contents of the sample CA key-ring file, `sslCAdefault.pfx`, by using iKeyman, to see if your personal certificates were signed by one of the listed CAs.

- If your personal certificates were signed by a listed CA, then you can use the sample CA key-ring file.
- If your personal certificates were signed by a different CA, you must create a key-ring file containing the public CA certificate of the CA that signed your personal certificates. This may have been returned with your personal certificate. If not, then you must request the CA certificate from the same CA that supplied your personal certificates and then add it to `sslCAdefault.pfx`.

If you need to add a CA certificate, you can use either the iKeyman CLI or the iKeyman GUI.

To add a CA certificate by using the iKeyman CLI:

```
mqiptKeycmd -cert -add -db sslCAdefault.pfx -pw key_password -type key_type
            -file ca_file_name.crt -label label
```

where:

- **`-db`** specifies the CA key-ring file, in this case `sslCAdefault.pfx`.
- **`-pw`** specifies the key-ring password.
- **`-type`** specifies the format of the key database; for example, `pkcs12`.
- **`-file`** specifies the name of the file returned by the CA.
- **`-label`** specifies a unique name of your choice; it is preferable not to use space characters.

To add a CA certificate by using the iKeyman GUI:

- In the Key Database Content panel, select `Signer Certificates` from the drop-down list
- Click **Add**.
- Enter the name of the file containing the CA certificate, then click **OK**.
- Enter a label for the CA certificate. The label can be any unique name you choose; it is preferable not to use space characters. Click **OK**.

To use these new key-ring files for server authentication, see the scenario Authenticating an SSL/TLS server, and set the following route properties:

```
SSLClientCAKeyRing=C:\\mqipt\\ssl\\sslCAdefault.pfx
SSLClientCAKeyRingPW=C:\\mqipt\\ssl\\sslCAdefault.pwd
SSLServerKeyRing=C:\\mqipt\\ssl\\myServer.pfx
SSLServerKeyRingPW=C:\\mqipt\\ssl\\myServer.pwd
SSLServerCAKeyRing=C:\\mqipt\\ssl\\sslCAdefault.pfx
SSLServerCAKeyRingPW=C:\\mqipt\\ssl\\sslCAdefault.pwd
```

To use these new key-ring files for client and server authentication, see the scenario SSL/TLS client authentication, and set the following route properties:

```
SSLClientKeyRing=C:\\mqipt\\ssl\\myClient.pfx
SSLClientKeyRingPW=C:\\mqipt\\ssl\\myClient.pwd
SSLClientCAKeyRing=C:\\mqipt\\ssl\\sslCAdefault.pfx
SSLClientCAKeyRingPW=C:\\mqipt\\ssl\\sslCAdefault.pwd
SSLServerKeyRing=C:\\mqipt\\ssl\\myServer.pfx
SSLServerKeyRingPW=C:\\mqipt\\ssl\\myServer.pwd
SSLServerCAKeyRing=C:\\mqipt\\ssl\\sslCAdefault.pfx
SSLServerCAKeyRingPW=C:\\mqipt\\ssl\\sslCAdefault.pwd
```

## Scenario: Creating test certificates

In this scenario, you can create a self-signed certificate which you can use for testing MQIPT routes. This 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 SSL/TLS connectivity without paying a Certificate Authority (CA) for a certificate. However, you should not use self-signed certificates in production environments. If you need certificates for production usage, see “Scenario: Creating a key-ring file” on page 72.

### Before you begin

- Before you start to use this scenario, make sure that you have completed the prerequisite tasks listed in Chapter 6, “Scenarios: Getting started with MQIPT,” on page 69.

### About this task

You can either use the iKeyman command-line interface (CLI) or the iKeyman GUI to request the certificate. You should then include the resulting key-ring file in the **SSLServerKeyRing** or **SSLClientKeyRing** MQIPT route property, depending on whether the certificate is for use by inbound or outbound connections.

### Procedure

Use one of the following methods to create test certificates:

- Use the iKeyman command-line interface (CLI)
  1. Create a new PKCS#12 file:

     ```
     mqiptKeycmd -keydb -create -db server_name.pfx -pw key_password -type key_type
     ```

     where:
     - **-db** specifies the file (*server_name*.pfx) that corresponds to the key-ring property of the MQIPT route. For example, use `SSLClientKeyRing` for an SSLClient route and `SSLServerKeyRing` for an SSLServer route.
     - **-pw** specifies the password (*key_password*) that you must use later with the **mqiptPW** utility to store the encrypted key-ring password.
     - **-type** specifies the format of the key database; for example, `pkcs12`.
  2. Create a self-signed personal certificate for testing purposes:

     ```
     mqiptKeycmd -cert -create -db server_name.pfx -pw password -type key_type
                 -label label -dn DN_identity
                 -sig_alg signature_algorithm -size key_size
     ```

     where:
     - **-type** specifies the format of the key database; for example, `pkcs12`.
     - **-label** specifies a unique name of your choice; it is preferable not to include space characters.
     - **-dn** specifies the appropriate Distinguished Name identity for the MQIPT route; for example, `"CN=Test Certificate,OU=Sales,O=Example,C=US"`.
     - **-sig_alg** specifies the hash algorithm; for example, `SHA256WithRSA`.
     - **-size** specifies the size of the public key; for example, `2048`.

If you use the example values given, 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.

When creating a certificate, take care to choose an appropriate public key encryption algorithm, key size, and digital signature algorithm for your organization's security needs. See "Digital certificate considerations for MQIPT" on page 38 for more information.

- Use the iKeyman GUI
  1. Open the iKeyman GUI by running the following command:
     ```
     mqiptKeyman
     ```
  2. Click **Key database file** > **New**.
  3. Select the type of the key database; for example, `PKCS12`.
  4. Enter the file name and location for the new key-ring file. This must correspond to the key-ring property of the MQIPT route; for example, use `SSLClientKeyRing` for an SSLClient route and `SSLServerKeyRing` for an SSLServer route. Click **OK**.
  5. Enter a password for the new key-ring file. Enter the password a second time to confirm. This is the password that you must use later with the **`mqiptPW`** utility to store the encrypted key-ring password. Click **OK** to create the new personal-certificate key-ring file.
  6. Create the new self-signed personal certificate by clicking **Create** > **New Self-Signed Certificate**.
  7. Enter a label for the new certificate in the **Key Label** field. The label can be any unique name you choose; it is preferable not to include space characters.
  8. Select the key size and digital signature algorithm as appropriate for your organization's security needs. See "Digital certificate considerations for MQIPT" on page 38 for more information.
  9. Enter the appropriate Distinguished Name identity for the MQIPT route in the optional DN fields, then click **OK**.

### What to do next

After you have finished configuring the key-ring file, store an encrypted copy of the password in a file so that MQIPT can access the key-ring file:

```
mqiptPW key_password password_file_name.pwd
```

where:

- *`key_password`* is the key-ring password.
- *`password_file_name`*`.pwd` is the name of the file to store the password in.

Ensure that this file is named by the appropriate MQIPT route property; for example, **`SSLServerKeyRingPW`** for the SSLServer personal certificate key-ring file or **`SSLClientKeyRingPW`** for the SSLClient personal certificate key-ring file.

## Scenario: Authenticating an SSL/TLS server

In this scenario, you can test an SSL/TLS connection by using the sample test certificate (`sslSample.pfx`) key-ring file, provided with MQIPT in the `ssl` subdirectory.

## Before you begin

- Before you start to use this scenario, make sure that you have completed the prerequisite tasks listed in Chapter 6, "Scenarios: Getting started with MQIPT," on page 69.

## About this task

The connection is made between a IBM MQ client and a IBM MQ server through two instances of MQIPT. During the SSL/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. (Based on `mqipt.conf` created from "Scenario: Verifying that MQIPT is working correctly" on page 70). For details on how to create a test certificate to use in this example, see "Scenario: Creating test certificates" on page 75.

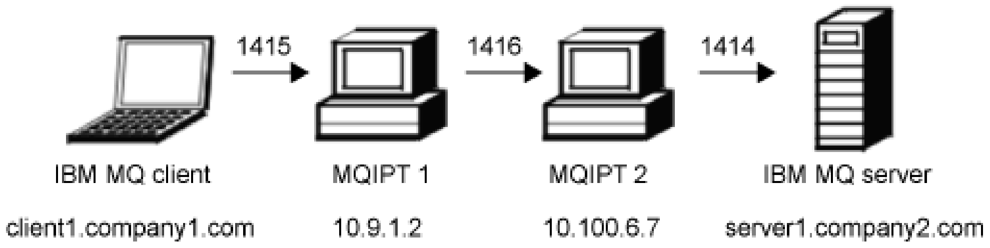


*Figure 7. SSL/TLS server network diagram*

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 an SSL/TLS server, complete the following steps:

1. On MQIPT1:
   a. Edit `mqipt.conf` and add a route definition:
      ```
      [route]
      ListenerPort=1415
      Destination=10.100.6.7
      DestinationPort=1416
      SSLClient=true
      SSLClientKeyRing=C:\mqipt\ssl\sslSample.pfx
      SSLClientKeyRingPW=C:\mqipt\ssl\sslSample.pwd
      SSLClientCipherSuites=SSL_RSA_WITH_AES_256_CBC_SHA256
      ```
   b. Open a command prompt and start MQIPT:
      ```
      C:\mqipt\bin\mqipt C:\mqiptHome
      ```

      where `C:\mqiptHome` indicates the location of the MQIPT configuration file, `mqipt.conf`.

      The following message indicates successful completion:

```
5639-L92 (C) Copyright IBM Corp. 2000, 2017 All Rights Reserved
MQCPI001 IBM MQ Internet
Pass-Thru Version 2.1.0.3 starting
MQCPI004 Reading configuration information from mqipt.conf
MQCPI021 Password checking has been enabled on the command port
MQCPI008 Listening for control commands on port 1881
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 protocols
MQCPI036 ....SSL Client side enabled with properties :
MQCPI139 ......secure socket protocols <NULL>
MQCPI031 ......cipher suites SSL_RSA_WITH_AES_256_CBC_SHA256
MQCPI032 ......keyring file C:\mqipt\ssl\sslSample.pfx
MQCPI047 ......CA keyring file <NULL>
MQCPI071 ......site certificate uses UID=*,CN=*,T=*,OU=*,DC=*,O=*,STREET=*,L=*,S
T=*,PC=*,C=*,DNQ=*
MQCPI038 ......peer certificate uses UID=*,CN=*,T=*,OU=*,DC=*,O=*,STREET=*,L=*,S
T=*,PC=*,C=*,DNQ=*
MQCPI078 Route 1415 ready for connection requests
```

2. On MQIPT2:
   a. Edit `mqipt.conf` and add a route definition:
   ```
   [route]
   ListenerPort=1416
   Destination=Server1.company2.com
   DestinationPort=1414
   SSLServer=true
   SSLServerKeyRing=C:\mqipt\ssl\sslSample.pfx
   SSLServerKeyRingPW=C:\mqipt\ssl\sslSample.pwd
   SSLServerCipherSuites=SSL_RSA_WITH_AES_256_CBC_SHA256
   ```
   b. Open a command prompt and start MQIPT:
   ```
   C:\mqipt\bin\mqipt C:\mqiptHome
   ```
   where `C:\mqiptHome` indicates the location of the MQIPT configuration file, `mqipt.conf`.

   The following message indicates successful completion:
   ```
   5639-L92 (C) Copyright IBM Corp. 2000, 2017 All Rights Reserved
   MQCPI001 IBM MQ Internet
   Pass-Thru Version 2.1.0.3 starting
   MQCPI004 Reading configuration information from mqipt.conf
   MQCPI021 Password checking has been enabled on the command port
   MQCPI008 Listening for control commands on port 1882
   MQCPI011 The path C:\mqiptHome\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 protocols
   MQCPI037 ....SSL Server side enabled with properties :
   MQCPI139 ......secure socket protocols <NULL>
   MQCPI031 ......cipher suites SSL_RSA_WITH_AES_256_CBC_SHA256
   MQCPI032 ......keyring file C:\mqipt\ssl\sslSample.pfx
   MQCPI047 ......CA keyring file <NULL>
   MQCPI071 ......site certificate uses UID=*,CN=*,T=*,OU=*,DC=*,O=*,STREET=*,L=*,S
   T=*,PC=*,C=*,DNQ=*
   MQCPI038 ......peer certificate uses UID=*,CN=*,T=*,OU=*,DC=*,O=*,STREET=*,L=*,S
   T=*,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.

## Scenario: Authenticating an SSL/TLS client

In this scenario, you can test an SSL/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 Chapter 6, “Scenarios: Getting started with MQIPT,” on page 69.

### About this task

During the SSL/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. (Based on `mqipt.conf` created from “Scenario: Verifying that MQIPT is working correctly” on page 70).

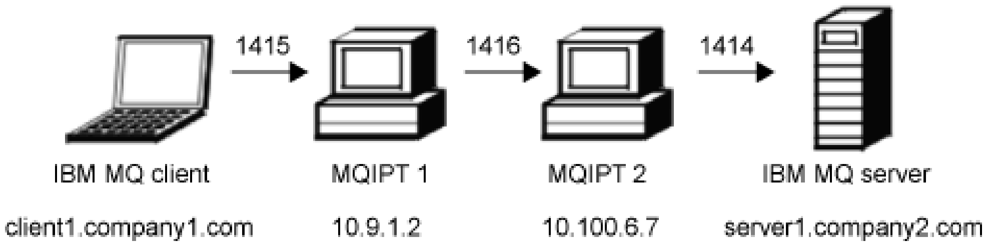


*Figure 8. SSL/TLS client network diagram*

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 an SSL/TLS client, complete the following steps:

1. On MQIPT1:
   a. Edit `mqipt.conf` and add a route definition:

```
[route]
ListenerPort=1415
Destination=10.100.6.7
DestinationPort=1416
SSLClient=true
SSLClientKeyRing=C:\mqipt\ssl\sslSample.pfx
SSLClientKeyRingPW=C:\mqipt\ssl\sslSample.pwd
SSLClientCipherSuites=SSL_RSA_WITH_AES_256_CBC_SHA256
```

   b. Open a command prompt and start MQIPT :

      ```
      C:\mqipt\bin\mqipt C:\mqiptHome
      ```

      where `C:\mqiptHome` indicates the location of the MQIPT configuration file, `mqipt.conf`.

      The following message indicates successful completion:

```
5639-L92 (C) Copyright IBM Corp. 2000, 2017 All Rights Reserved
MQCPI001 IBM MQ Internet
Pass-Thru Version 2.1.0.3 starting
MQCPI004 Reading configuration information from mqipt.conf
MQCPI021 Password checking has been enabled on the command port
MQCPI008 Listening for control commands on port 1881
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 protocols
MQCPI036 ....SSL Client side enabled with properties :
MQCPI139 ......secure socket protocols <NULL>
MQCPI031 ......cipher suites SSL_RSA_WITH_AES_256_CBC_SHA256
MQCPI032 ......keyring file C:\mqipt\ssl\sslSample.pfx
MQCPI047 ......CA keyring file <NULL>
MQCPI071 ......site certificate uses UID=*,CN=*,T=*,OU=*,DC=*,O=*,STREET=*,L=*,S
T=*,PC=*,C=*,DNQ=*
MQCPI038 ......peer certificate uses UID=*,CN=*,T=*,OU=*,DC=*,O=*,STREET=*,L=*,S
T=*,PC=*,C=*,DNQ=*
MQCPI078 Route 1415 ready for connection requests
```

2. On MQIPT2:
   a. Edit `mqipt.conf` and add a route definition:

```
[route]
ListenerPort=1416
Destination=Server1.company2.com
DestinationPort=1414
SSLServer=true
SSLServerAskClientAuth=true
SSLServerKeyRing=C:\mqipt\ssl\sslSample.pfx
SSLServerKeyRingPW=C:\mqipt\ssl\sslSample.pwd
SSLServerCipherSuites=SSL_RSA_WITH_AES_256_CBC_SHA256
```

   b. Open a command prompt and start MQIPT:

      ```
      C:\mqipt\bin\mqipt C:\mqiptHome
      ```

      where `C:\mqiptHome` indicates the location of the MQIPT configuration file, `mqipt.conf`.

      The following message indicates successful completion:

```
5639-L92 (C) Copyright IBM Corp. 2000, 2017 All Rights Reserved
MQCPI001 IBM MQ Internet
Pass-Thru Version 2.1.0.3 starting
MQCPI004 Reading configuration information from mqipt.conf
MQCPI021 Password checking has been enabled on the command port
MQCPI008 Listening for control commands on port 1882
MQCPI011 The path C:\mqiptHome\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 protocols
```

```
MQCPI037 ....SSL Server side enabled with properties :
MQCPI139 ......secure socket protocols <NULL>
MQCPI031 ......cipher suites SSL_RSA_WITH_AES_256_CBC_SHA256
MQCPI032 ......keyring file C:\mqipt\ssl\sslSample.pfx
MQCPI047 ......CA keyring file <NULL>
MQCPI071 ......site certificate uses UID=*,CN=*,T=*,OU=*,DC=*,O=*,STREET=*,L=*,S
T=*,PC=*,C=*,DNQ=*
MQCPI038 ......peer certificate uses UID=*,CN=*,T=*,OU=*,DC=*,O=*,STREET=*,L=*,S
T=*,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, 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.

## Scenario: 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 Chapter 6, “Scenarios: Getting started with MQIPT,” on page 69.

### About this task

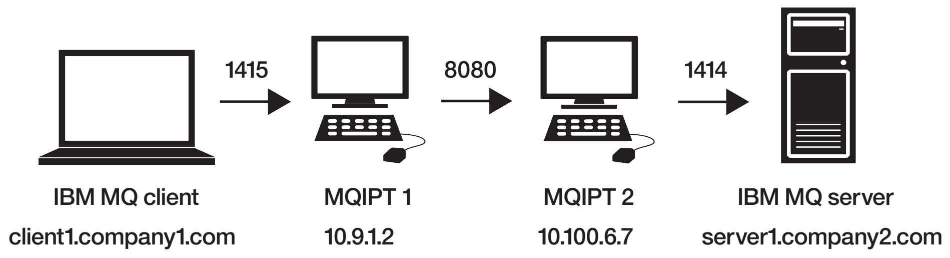


*Figure 9. HTTP proxy network diagram*

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 an HTTP proxy, complete the following steps:

1. On MQIPT1:
   a. Edit `mqipt.conf` and add a 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
      ```
      where `C:\mqiptHome` indicates the location of the MQIPT configuration file, `mqipt.conf`.

      The following message indicates successful completion:
      ```
      5639-L92 (C) Copyright IBM Corp. 2000, 2017 All Rights Reserved
      MQCPI001 IBM MQ Internet
      Pass-Thru Version 2.1.0.4 starting
      MQCPI004 Reading configuration information from mqipt.conf
      MQCPI021 Password checking has been enabled on the command port
      MQCPI008 Listening for control commands on port 1881
      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 MQIPT2:
   a. Edit `mqipt.conf` and add a route definition:
      ```
      [route]
      ListenerPort=8080
      Destination=Server1.company2.com
      DestinationPort=1414
      ```
   b. Open a command prompt and start MQIPT:
      ```
      C:\mqipt\bin\mqipt C:\mqiptHome
      ```
      where `C:\mqiptHome` indicates the location of the MQIPT configuration file, `mqipt.conf`.

      The following message indicates successful completion:
      ```
      5639-L92 (C) Copyright IBM Corp. 2000, 2017 All Rights Reserved
      MQCPI001 IBM MQ Internet
      Pass-Thru Version 2.1.0.4 starting
      MQCPI004 Reading configuration information from mqipt.conf
      MQCPI021 Password checking has been enabled on the command port
      MQCPI008 Listening for control commands on port 1881
      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
      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.

## Scenario: Configuring an HTTP proxy

In this scenario, you can test the connection using an HTTP proxy (IBM Caching Proxy).

### Before you begin

- Before you start to use this scenario, make sure that you have completed the prerequisite tasks listed in Chapter 6, "Scenarios: Getting started with MQIPT," on page 69.
- Check the following properties in the `ibmproxy.conf` file:
  - **`ProxyPersistence`** must be set to `true` to permit persistent connections.
  - **`MaxPersistRequest`** must be set to `5000`, the number of requests allowed on a single connection before the connection is broken.
  - **`PersistTimeout`** must be set to `12`, the time (in hours) allowed for the connection to exist.
  - **`Proxy`** entries must include the **`USESESSION`** parameter.

### About this task

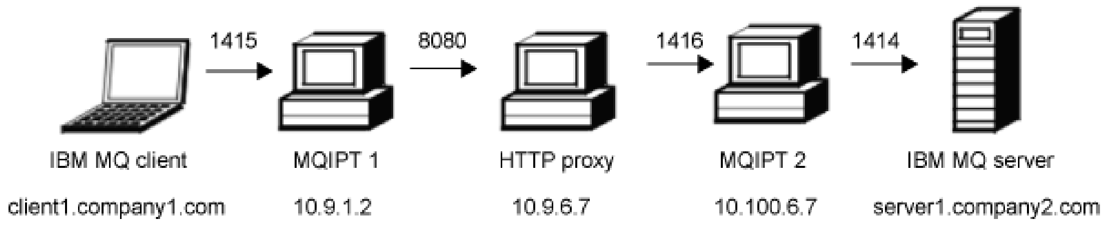


*Figure 10. HTTP proxy network diagram*

This diagram shows the connection from the IBM MQ client (called client1.company1.com on port 1415) through MQIPT1, through the HTTP proxy computer (on port 1080), through MQIPT2, and finally to the IBM MQ server (called server1.company2.com on port 1414).

### Procedure

To configure an HTTP proxy, complete the following steps:

1. On MQIPT1:
   a. Edit `mqipt.conf` and add a route definition:

```
[route]
ListenerPort=1415
Destination=10.100.6.7
DestinationPort=1416
HTTP=true
HTTPProxyPort=8080
HTTPProxy=10.9.6.7
```

   b. Open a command prompt and start MQIPT:

   ```
   C:\mqipt\bin\mqipt C:\mqiptHome
   ```

   where `C:\mqiptHome` indicates the location of the MQIPT configuration file, `mqipt.conf`.

   The following message indicates successful completion:

   ```
   5639-L92 (C) Copyright IBM Corp. 2000, 2017 All Rights Reserved
   MQCPI001 IBM MQ Internet
   Pass-Thru Version 2.1.0.3 starting
   MQCPI004 Reading configuration information from C:\mqiptHome\mqipt\mqipt.conf
   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 ....9.100.6.7(1416)
   MQCPI035 ....using HTTP
   MQCPI024 ....and HTTP proxy at 10.9.6.7(8080)
   MQCPI078 Route 1415 ready for connection requests
   ```

2. On MQIPT2:
   a. Edit `mqipt.conf` and add a route definition:

   ```
   [route]
   ListenerPort=1416
   Destination=Server1.company2.com
   DestinationPort=1414
   ```

   b. Open a command prompt and start MQIPT:

   ```
   C:
   cd \mqipt\bin
   mqipt ..
   ```

   where `..` indicates that the MQIPT configuration file, `mqipt.conf`, is in the parent directory.

   The following message indicates successful completion:

   ```
   5639-L92 (C) Copyright IBM Corp. 2000, 2017 All Rights Reserved
   MQCPI001 IBM MQ Internet
   Pass-Thru Version 2.1.0.3 starting
   MQCPI004 Reading configuration information from C:\mqiptHome\mqipt.conf
   MQCPI011 The path C:\mqiptHome\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 protocols
   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.

## Scenario: 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 Chapter 6, "Scenarios: Getting started with MQIPT," on page 69.

### About this task

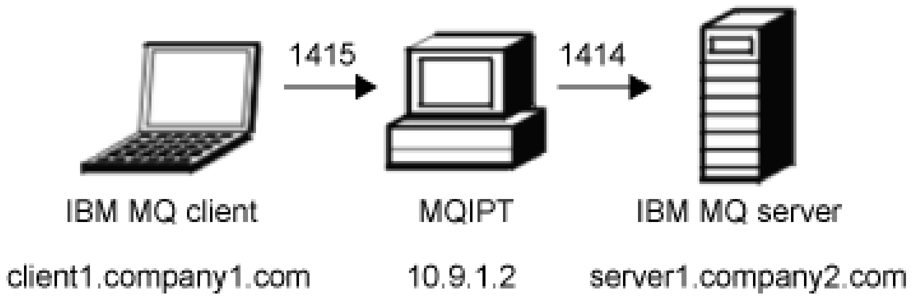


*Figure 11. Access control network diagram*

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\ssl\mqiptSample.policy C:\mqiptHome\mqipt.policy
   ```
   b. Add a policy definition 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**: `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`. Add two properties to the `[global]` section:

```
SecurityManager=true
SecurityManagerPolicy=C:\mqiptHome\mqipt.policy
```

Add a 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
```

where `C:\mqiptHome` indicates the location of the MQIPT configuration file, `mqipt.conf`.

The following message indicates successful completion:

```
5639-L92 (C) Copyright IBM Corp. 2000, 2017 All Rights Reserved
MQCPI001 IBM MQ Internet
Pass-Thru Version 2.1.0.3 starting
MQCPI004 Reading configuration information from C:\mqiptHome\mqipt.conf
MQCPI055 Setting the java.security.policy to C:\mqiptHome\mqipt.policy
MQCPI053 Starting the Java Security Manager
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 protocols
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.

## Scenario: 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 Chapter 6, "Scenarios: Getting started with MQIPT," on page 69.
- Enable SOCKS on either the whole IBM MQ computer or just the IBM MQ client application (amqsputc/amqsgetc).
- Configure the SOCKS client as follows:
  1. Use MQIPT as the SOCKS proxy.
  2. Enable SOCKS Version 5 support.
  3. Disable user authentication.
  4. Restrict connections to the MQIPT network address.

  (The values of the MQIPT **`Destination`** and **`DestinationPort`** properties can be anything, as the true destination is obtained from the IBM MQ client during the SOCKS handshaking process.)

### About this task

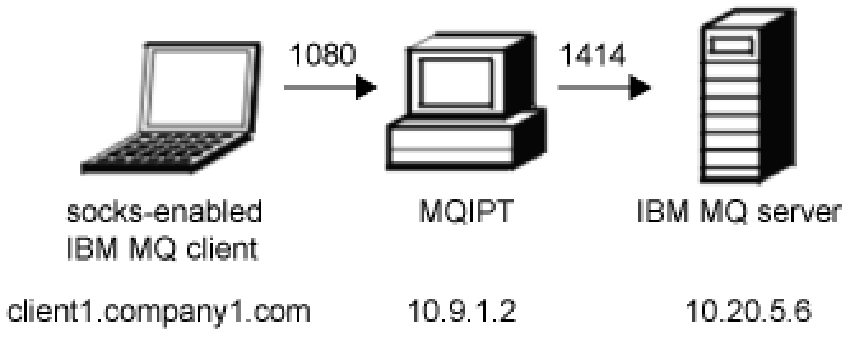


*Figure 12. SOCKS proxy network diagram*

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. On MQIPT1:
   a. Edit `mqipt.conf` and add a route definition:

```
[route]
ListenerPort=1080
Destination=server1.company2.com
DestinationPort=1414
SocksServer=true
```

b. Open a command prompt and start MQIPT:

```
C:\mqipt\bin\mqipt C:\mqiptHome
```

where `C:\mqiptHome` indicates the location of the MQIPT configuration file, `mqipt.conf`.

The following message indicates successful completion:

```
5639-L92 (C) Copyright IBM Corp. 2000, 2017 All Rights Reserved
MQCPI001 IBM MQ Internet
Pass-Thru Version 2.1.0.3 starting
MQCPI004 Reading configuration information from C:\mqiptHome\mqipt.conf
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 protocols
MQCPI052 ....Socks server side enabled
MQCPI078 Route 1080 ready for connection requests
```

2. 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/10.20.5.6(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.

## Scenario: 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, 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 Chapter 6, “Scenarios: Getting started with MQIPT,” on page 69.
-

## About this task

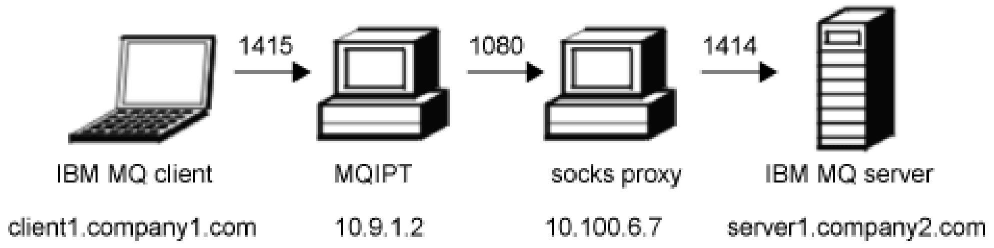


*Figure 13. SOCKS client network diagram*

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
   ```

   where `C:\mqiptHome` indicates the location of the MQIPT configuration file, `mqipt.conf`.

   The following message indicates successful completion:

   ```
   5639-L92 (C) Copyright IBM Corp. 2000, 2017 All Rights Reserved
   MQCPI001 IBM MQ Internet
   Pass-Thru Version 2.1.0.3 starting
   MQCPI004 Reading configuration information from C:\mqiptHome\mqipt.conf
   MQCPI022 Password checking has been disabled 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 protocols
   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.

## Scenario: 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 Chapter 6, “Scenarios: Getting started with MQIPT,” on page 69.
- 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

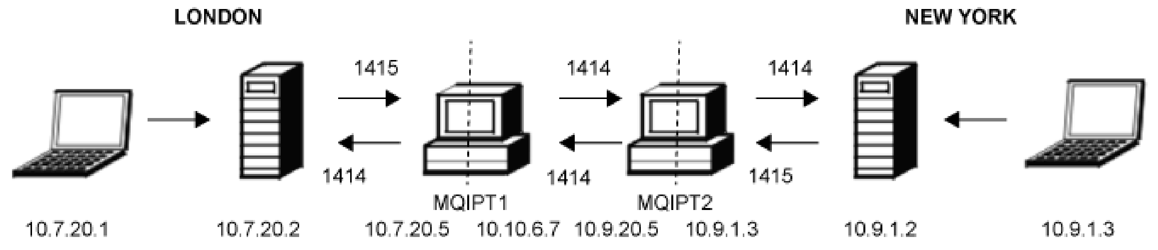


*Figure 14. Clustering network diagram*

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 MQIPT1. Edit `mqipt.conf` and add two 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 MQIPT1. Open a command prompt and enter:

   ```
   C:\mqipt\bin\mqipt C:\mqiptHome
   ```

   where `C:\mqiptHome` indicates the location of the MQIPT configuration file, `mqipt.conf`.

   The following message indicates successful completion:

   ```
   5639-L92 (C) Copyright IBM Corp. 2000, 2017 All Rights Reserved
   MQCPI001 IBM MQ Internet
   Pass-Thru Version 2.1.0.0 starting
   ```

```
MQCPI004 Reading configuration information from C:\mqiptHome\mqipt.conf
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 ....11.9.20.5(1414)
MQCPI035 ....using MQ protocols
MQCPI052 ....Socks server side enabled
MQCPI078 Route 1415 ready for connection requests
MQCPI006 Route 1414 has started and will forward messages to :
MQCPI034 ....8.7.20.2(1414)
MQCPI035 ....using MQ protocols
MQCPI078 Route 1414 ready for connection requests
```

5. Set up MQIPT2. Edit `mqipt.conf` and add two 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 MQIPT2. Open a command prompt and enter the following commands:

   ```
   C:
   cd \mqipt\bin
   mqipt ..
   ```

   where `..` indicates that the MQIPT configuration file, `mqipt.conf`, is in the parent directory.

   The following message indicates successful completion:

   ```
   5639-L92 (C) Copyright IBM Corp. 2000, 2017 All Rights Reserved
   MQCPI001 IBM MQ Internet
   Pass-Thru Version 2.1.0.0 starting
   MQCPI004 Reading configuration information from C:\mqipt\mqipt.conf
   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 protocols
   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 protocols
   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.

## Scenario: 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 Chapter 6, "Scenarios: Getting started with MQIPT," on page 69.
- Install MQIPT on a multihomed computer.

### About this task

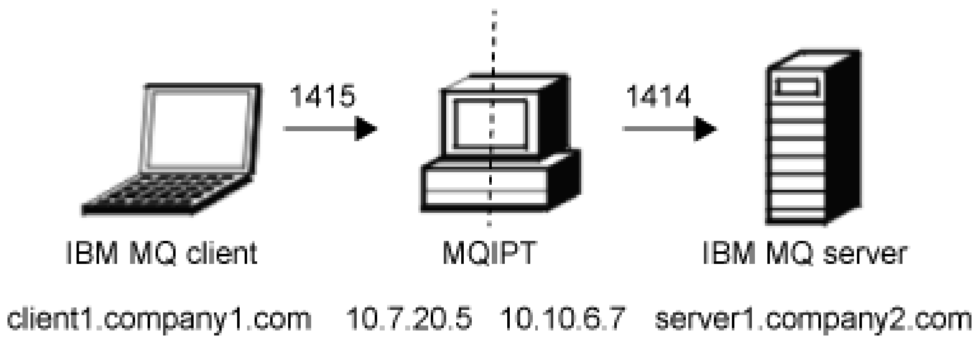


*Figure 15. Port allocation network diagram*

This diagram shows the connection from a IBM MQ client (client1.company1.com on port 1415) through MQIPT to a 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 a 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 client, and enter the following command:

   ```
   C:\mqipt\bin\mqipt C:\mqiptHome
   ```

   where `C:\mqiptHome` indicates the location of the MQIPT configuration file, `mqipt.conf`.

   The following message indicates successful completion:

   ```
   5639-L92 (C) Copyright IBM Corp. 2000, 2017 All Rights Reserved
   MQCPI001 IBM MQ Internet
   Pass-Thru Version 2.1.0.3 starting
   MQCPI004 Reading configuration information from mqipt.conf
   MQCPI021 Password checking has been enabled on the command port
   MQCPI008 Listening for control commands on port 1881
   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 protocols
   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, 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.

## Scenario: 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 Chapter 6, “Scenarios: Getting started with MQIPT,” on page 69.
- Ensure that MQIPT2 has a personal certificate, issued by the trusted Certificate Authority (CA), stored in a key-ring file called `myCert.pfx` and the encrypted password used to open the key-ring file is stored in the file `myCert.pwd`.
- Ensure that MQIPT1 has a copy of the trusted CA certificate that will be used to authenticate the certificate sent by IPT2. This certificate is stored in a key-ring file called `caCerts.pfx` and the encrypted password used to open the key-ring file is stored in the file `caCerts.pwd`.
- The encrypted password files have been created by using the mqiptPW script.

## About this task

In this scenario, you can connect the IBM MQ client to a queue manager (QM) and place a IBM MQ message on the target queue. Running an MQIPT trace on MQIPT1 will show the LDAP server being used.

To demonstrate how CRLs work, make sure that the personal certificate used by MQIPT2 is revoked by the trusted CA. Then the IBM MQ client is not allowed to connect to the QM, as the connection from MQIPT1 to MQIPT2 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.

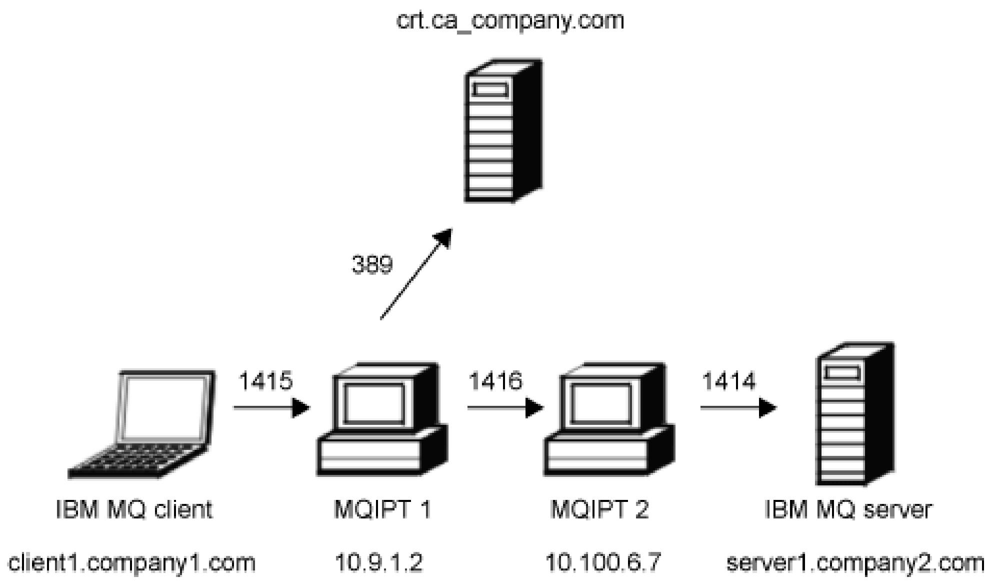


*Figure 16. LDAP server network diagram*

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 MQIPT1:
   a. Edit `mqipt.conf` and add a route definition:

```
[route]
ListenerPort=1415
Destination=10.100.6.7
DestinationPort=1416
SSLClient=true
SSLClientCAKeyRing=C:\mqipt\ssl\caCerts.pfx
SSLClientCAKeyRingPW=C:\mqipt\ssl\caCerts.pwd
LDAP=true
LDAPServer1=crl.ca_company.com
LDAPServer1Timeout=4
```

b. Open a command prompt and start MQIPT:

```
C:\mqipt\bin\mqipt C:\mqiptHome
```

where `C:\mqiptHome` indicates the location of the MQIPT configuration file, `mqipt.conf`.

The following message indicates successful completion:

```
5639-L92 (C) Copyright IBM Corp. 2000, 2017 All Rights Reserved
MQCPI001 IBM MQ Internet
Pass-Thru Version 2.1.0.3 starting
MQCPI004 Reading configuration information from C:\mqiptHome\mqipt.conf
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 protocols
MQCPI036 ....SSL Client side enabled with properties :
MQCPI031 ......CipherSuites <NULL>
MQCPI032 ......keyring file <NULL>
MQCPI047 ......CA keyring file C:\mqipt\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 MQIPT2:
   a. Edit `mqipt.conf` and add a route definition:

```
[route]
ListenerPort=1416
Destination=server1.company2.com
DestinationPort=1414
SSLServer=true
SSLServerKeyRing=C:\mqipt\ssl\myCert.pfx
SSLServerKeyRingPW=C:\mqipt\ssl\myCert.pwd
```

   b. Open a command prompt and start MQIPT:

```
C:
cd \mqipt\bin
mqipt ..
```

where `..` indicates that the MQIPT configuration file, `mqipt.conf`, is in the parent directory.

The following message indicates successful completion:

```
5639-L92 (C) Copyright IBM Corp. 2000, 2017 All Rights Reserved
MQCPI001 IBM IBM MQ Internet
Pass-Thru Version 2.1.0.3 starting
MQCPI004 Reading configuration information from C:\mqipt\mqipt.conf
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 protocols
MQCPI037 ....SSL Server side enabled with properties :
MQCPI031 ......CipherSuites <NULL>
MQCPI032 ......keyring file C:\mqipt\ssl\myCert.pfx
MQCPI047 ......CA keyring 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.

## Scenario: running MQIPT in SSL/TLS proxy mode

You can run MQIPT in SSL/TLS proxy mode, so that it accepts an SSL/TLS connection request from an IBM MQ SSL/TLS client and tunnels it to a IBM MQ SSL/TLS server.

### Before you begin

- Before you start to use this scenario, make sure that you have completed the prerequisite tasks listed in Chapter 6, “Scenarios: Getting started with MQIPT,” on page 69.
- Set up the IBM MQ client and queue manager to use an SSL/TLS channel.
- Configure the IBM MQ client and server to use an SSL/TLS connection.

## About this task

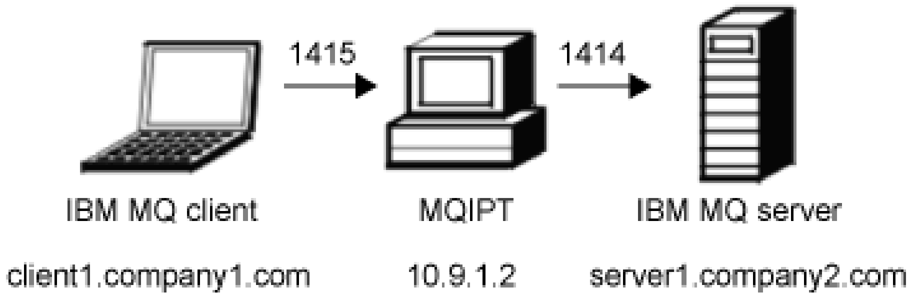


*Figure 17. SSL/TLS proxy mode network diagram*

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 setting up SSL/TLS for IBM MQ, refer to the *Security* section of the IBM MQ product documentation.

## Procedure

To run MQIPT in SSL/TLS proxy mode, complete the following steps:

1. Edit `mqipt.conf` and add a new route definition:
   ```
   [route]
   ListenerPort=1415
   Destination=server1.company2.com
   DestinationPort=1414
   SSLProxyMode=true
   ```
2. Start MQIPT. Open a command prompt and enter the following command:
   ```
   C:\mqipt\bin\mqipt C:\mqiptHome
   ```

   where `C:\mqiptHome` indicates the location of the MQIPT configuration file, `mqipt.conf`.

   The following message indicates successful completion:
   ```
   5639-L92 (C) Copyright IBM Corp. 2000, 2017 All Rights Reserved
   MQCPI001 IBM MQ Internet
   Pass-Thru Version 2.1.0.3 starting
   MQCPI004 Reading configuration information from C:\mqiptHome\mqipt.conf
   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
   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.

## Scenario: running MQIPT in SSL/TLS proxy mode with a security manager

You can run MQIPT in SSL/TLS proxy mode, so that it accepts an SSL/TLS connection request from an IBM MQ SSL/TLS client and tunnels it to a IBM MQ SSL/TLS server. By using a security manager with MQIPT, you can restrict the addresses to which messages can be sent.

### Before you begin

- Before you start to use this scenario, make sure that you have completed the prerequisite tasks listed in Chapter 6, “Scenarios: Getting started with MQIPT,” on page 69.
- Set up the IBM MQ client and queue manager to use an SSL/TLS channel.
- Configure the IBM MQ client and server to use an SSL/TLS connection.

### About this task

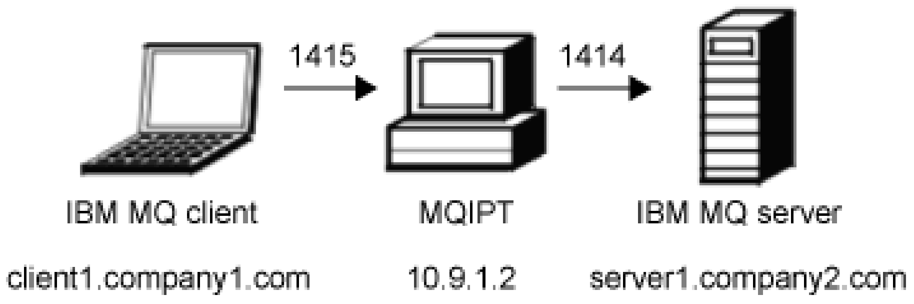


*Figure 18. SSL/TLS proxy mode network diagram*

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 setting up SSL/TLS for IBM MQ, refer to the *Security* section of the IBM MQ product documentation.

## Procedure

To run MQIPT in SSL/TLS proxy mode with a security manager, complete the following steps:

1. 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\ssl\mqiptSample.policy C:\mqiptHome\mqipt.policy
   ```
2. Add a policy definition by using the following command:

   ```
   C:\mqipt\java\jre\bin\policytool
   ```

   In the policy tool:
   a. Click **File** > **Open** > **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**: `SocketPermission`
      **Target**: `client1.company1.com:1024-`
      **Actions**: `accept, listen, resolve`
   g. Click **File > Save** to save the changes to the policy file.
3. Edit `mqipt.conf` and add the following properties to the `[global]` section and add a new route definition:

   ```
   [global]

   SecurityManager=true
   SecurityManagerPolicy=C:\mqiptHome\mqipt.policy

   [route]
   ListenerPort=1415
   Destination=server1.company2.com
   DestinationPort=1414
   SSLProxyMode=true
   ```
4. Start MQIPT. Open a command prompt, and enter the following command:

   ```
   C:\mqipt\bin\mqipt C:\mqiptHome
   ```

where C:\mqiptHome indicates the location of the MQIPT configuration file, mqipt.conf.

The following message indicates successful completion:

```
5639-L92 (C) Copyright IBM Corp. 2000, 2017 All Rights Reserved
MQCPI001 IBM MQ Internet
Pass-Thru Version 2.1.0.3 starting
MQCPI004 Reading configuration information from C:\mqiptHome\mqipt\mqipt.conf
MQCPI055 Setting the java.security.policy to C:\mqiptHome\mqipt.policy
MQCPI053 Starting the Java Security Manager
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
MQCPI078 Route 1415 ready for connection requests
```

5. 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.

## Scenario: Apache rewrite

In this scenario, you can use the rewrite directive to convert an HTTP request into an internal Apache proxy redirect.

The proxy and rewrite modules must be loaded, but as Apache is not really working in proxy mode, all proxy directives can remain commented out.

### Before you begin

- Before you start to use this scenario, make sure that you have completed the prerequisite tasks listed in Chapter 6, “Scenarios: Getting started with MQIPT,” on page 69.
- Install an Apache HTTP server in C:\apache. (If it is installed elsewhere, adjust the paths in the following steps accordingly.)
- Install IBM Caching Proxy in C:\cp\etc\en_US. (If it is installed elsewhere, adjust the paths in the following steps accordingly.)

## About this task

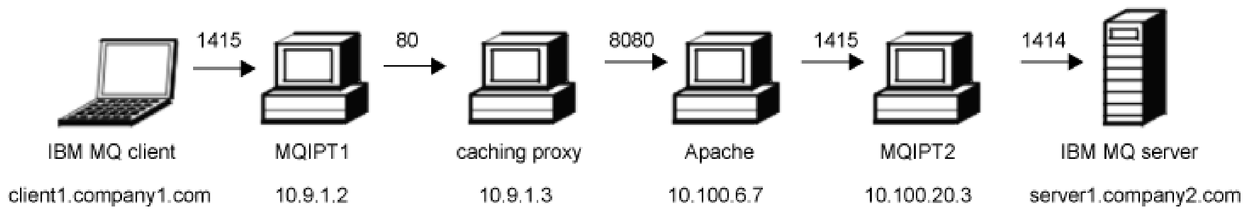


*Figure 19. Apache rewrite network diagram*

This diagram shows the connection from the IBM MQ client (called client1.company1.com on port 1415) through MQIPT, Caching Proxy, Apache, MQIPT, and finally to the IBM MQ server (called server1.company2.com on port 1414).

## Procedure

To use the rewrite directive, complete the following steps:

1. On On WTE: Edit `C:\cp\etc\en_US\ibmproxy.conf` and change the following properties:

   ```
   ProxyPersistence ON
   MaxPersistRequest  5000
   ```

2. On Apache: Edit `C:\apache\conf\httpd.conf`:

   ```
   RewriteEngine on
   RewriteLog logs/rewrite.log
   RewriteLogLevel 9
   RewriteRule ^/mqipt  http://%{HTTP:Host}/mqipt  [P]

   LoadModule proxy_module modules/mod_proxy.so
   LoadModule proxy_connect_module modules/mod_proxy_connect.so
   LoadModule proxy_http_module modules/mod_proxy_http.so
   LoadModule negotiation_module modules/mod_negotiation.so
   LoadModule rewrite_module modules/mod_rewrite.so

   start Apache
   ```

3. On MQIPT1:
   a. Edit `mqipt.conf` and add a route definition:

      ```
      [route]
      ListenerPort=1415
      Destination=server1.company2.com
      DestinationPort=1414
      HTTP=true
      HTTPProxy=10.9.1.3
      HTTPProxyPort=80
      HTTPServer=10.100.6.7
      HTTPServerPort=8080
      ```

   b. Open a command prompt and start MQIPT:

      ```
      C:\mqipt\bin\mqipt C:\mqiptHome
      ```

      where `C:\mqiptHome` indicates the location of the MQIPT configuration file, `mqipt.conf`.

The following message indicates successful completion:

```
5639-L92 (C) Copyright IBM Corp. 2000, 2017 All Rights Reserved
MQCPI001 IBM MQ Internet
Pass-Thru Version 2.1.0.0 starting
MQCPI004 Reading configuration information from C:\mqiptHome\mqipt.conf
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 HTTP
MQCPI024 ....and HTTP proxy at 10.9.1.3(80)
MQCPI066 ....and HTTP server at 10.100.6.7(8080)
MQCPI078 Route 1415 ready for connection requests
```

4. On MQIPT2:
   a. Edit `mqipt.conf` and add a route definition:
      ```
      [route]
      ListenerPort=1415
      Destination=server1.company2.com
      DestinationPort=1414
      ```
   b. Open a command prompt and start MQIPT:
      ```
      C:
      cd \mqipt\bin
      mqipt ..
      ```
      (.. indicates that the MQIPT configuration file, `mqipt.conf`, is in the parent directory.)

      The following message indicates successful completion:
      ```
      5639-L92 (C) Copyright IBM Corp. 2000, 2017 All Rights Reserved
      MQCPI001 IBM MQ Internet
      Pass-Thru Version 2.1.0.0 starting
      MQCPI004 Reading configuration information from C:\mqipt\mqipt.conf
      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 ....server1.company2.com(1414)
      MQCPI035 ....using MQ protocols
      MQCPI078 Route 1415 ready for connection requests
      ```
5. 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.

## Scenario: 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 Chapter 6, “Scenarios: Getting started with MQIPT,” on page 69.
- Install Java 8.0 JDK.
- Add the Java `bin` subdirectory to the **PATH** environment variable.

## About this task

If you use the suggested srvconn channel name of `MQIPT.CONN.CHANNEL` (as used in most of these scenarios), the client connection will be allowed to complete and a IBM MQ message can be placed on the queue.

To demonstrate that the security exit is working as expected, define another srvconn channel with any name that does not start with the characters `MQIPT.` (for example, `TEST.CONN.CHANNEL`) and try the **amqsputc** 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 error will be given.

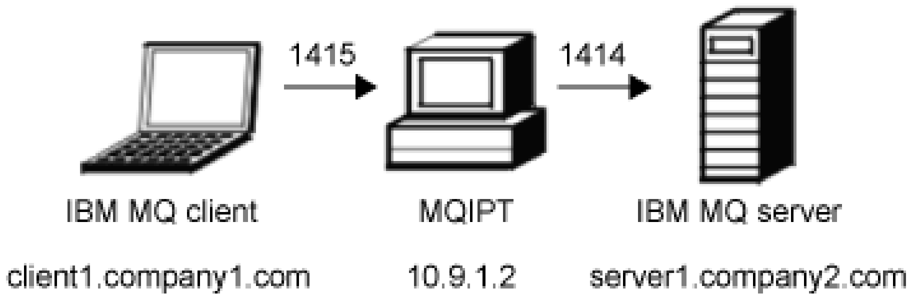


*Figure 20. Security exit network diagram*

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 MQIPT1:
   a. Open a command prompt and enter the following commands:
   ```
   C:
   cd \mqipt\exits
   javac -classpath C:\mqipt\lib\com.ibm.mq.ipt.jar;. SampleSecurityExit.java
   ```
   b. Edit `mqipt.conf` and add a route definition:

```
[route]
ListenerPort=1415
Destination=server1.company2.com
DestinationPort=1414
SecurityExit=true
SecurityExitName=SampleSecurityExit
```

c. Open a command prompt and start MQIPT:

```
C:\mqipt\bin\mqipt C:\mqiptHome
```

where `C:\mqiptHome` indicates the location of the MQIPT configuration file, `mqipt.conf`.

The following message indicates successful completion:

```
5639-L92 (C) Copyright IBM Corp. 2000, 2017 All Rights Reserved
MQCPI001 IBM MQ Internet
Pass-Thru Version 2.1.0.3 starting
MQCPI004 Reading configuration information from C:\mqiptHome\mqipt.conf
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 protocols
MQCPI079 ....using security exit C:\mqipt\exits\SampleSecurityExit
MQCPI080 ......and timeout of 5 seconds
MQCPI078 Route 1415 ready for connection requests
```

2. 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.

## Scenario: 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 Chapter 6, “Scenarios: Getting started with MQIPT,” on page 69.
- Install Java 8.0 JDK.
- Add the Java `bin` subdirectory to the **PATH** environment variable.

### About this task

The list of server names to be used is read from a configuration file. The name and location of the configuration file, called `SampleRoutingExit.conf`, is defined with the MQIPT **SecurityExitName** and **SecurityExitPath** properties.

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. But 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.

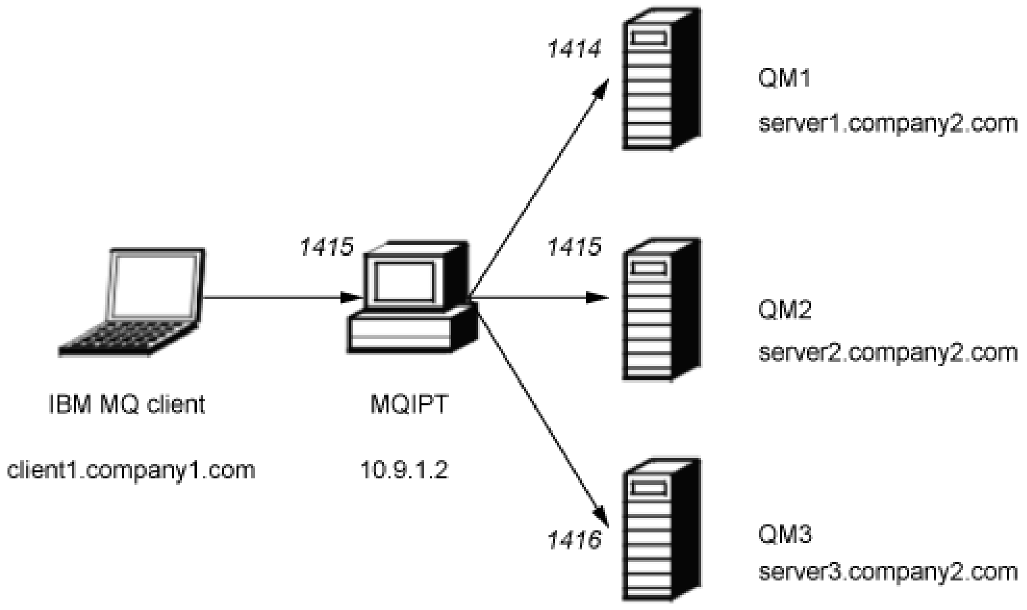


*Figure 21. Routing security exit network diagram*

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 MQIPT:

a. In the `C\mqiptHome\exits` subdirectory (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:

   where

   ```
   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.
b. Open a command prompt and enter the following command to copy the sample exit `SampleRoutingExit.java` to your MQIPT home directory:

   ```
   copy C:\mqipt\exits\SampleRoutingExit.java C:\mqiptHome\exits\SampleRoutingExit.java
   ```
c. Enter the following commands to compile the sample exit:

   ```
   C:
   cd \mqiptHome\exits
   javac -classpath C:\mqipt\lib\com.ibm.mq.ipt.jar;. SampleRoutingExit.java
   ```
d. 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: You do not necessarily have to set **`SecurityExitPath`** if you put `SampleRoutingExit.conf` in the default `exits` subdirectory as described in step 2b.
e. Start MQIPT. Open a command prompt and enter the following command:

   ```
   C:\mqipt\bin\mqipt C:\mqiptHome
   ```

   where `C:\mqiptHome` indicates the location of the MQIPT configuration file, `mqipt.conf`.

   The following message indicates successful completion:

   ```
   5639-L92 (C) Copyright IBM Corp. 2000, 2017 All Rights Reserved
   MQCPI001 IBM MQ Internet
   Pass-Thru Version 2.1.0.0 starting
   MQCPI004 Reading configuration information from C:\mqiptHome\mqipt.conf
   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 protocols
   MQCPI079 ....using security exit C:\mqiptHome\exits\SampleRoutingExit
   MQCPI080 ......and timeout of 5 seconds
   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 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.

## Scenario: 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 Chapter 6, “Scenarios: Getting started with MQIPT,” on page 69.
- 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, MQIPT need use only one route to service all connection requests. For example, to connect to QM1, the name of a SVRCONN channel might be QM1.MQIPT.CHANNEL.

The list of queue manager and server names to be used is read from a configuration file. The name and location of the configuration file, called `SampleOneRouteExit.conf`, is defined with the MQIPT **SecurityExitName** and **SecurityExitPath** properties.

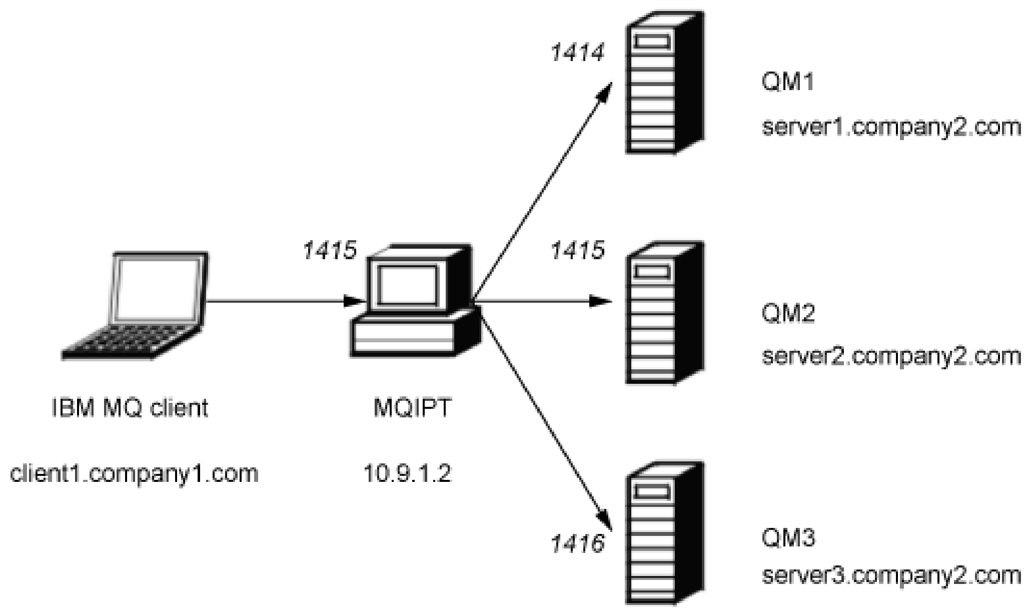


*Figure 22. Dynamic one route exit network diagram*

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 MQIPT1:
   a. In the *mqipt_path*\exits subdirectory (where *mqipt_path* is the location where MQIPT is installed), create a sample configuration file, called `SampleRoutingExit.conf` that contains the names of your three queue managers:

      ```
      QM1   server1.company2.com:1414
      QM2   server2.company2.com:1415
      QM3   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.

      You must change these server 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.

b. Open a command prompt and enter the following commands:
```
C:
cd \mqipt\exits
javac -classpath C:\mqipt\lib\com.ibm.mq.ipt.jar;. SampleOneRouteExit.java
```
c. Edit `mqipt.conf` and add a route definition:
```
[route]
ListenerPort=1415
Destination=server1.company2.com
DestinationPort=1414
SecurityExit=true
SecurityExitName=SampleOneRouteExit
```
d. Open a command prompt and start MQIPT:
```
C:\mqipt\bin\mqipt C:\mqiptHome
```
where `C:\mqiptHome` indicates the location of the MQIPT configuration file, `mqipt.conf`.

The following message indicates successful completion:
```
5639-L92 (C) Copyright IBM Corp. 2000, 2017 All Rights Reserved
MQCPI001 IBM MQ Internet
Pass-Thru Version 2.1.0.0 starting
MQCPI004 Reading configuration information from C:\mqiptHome\mqipt.conf
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 protocols
MQCPI079 ....using security exit C:\mqipt\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, 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.

c. Get the message:
```
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.

f. Get the message:
```
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.

i. Get the message:

```
amqsgetc MQIPT.LOCAL.QUEUE QM3
```

The message, `Hello world 3` is returned.

## Scenario: Using a certificate exit to authenticate an SSL/TLS server

In this scenario, you can authenticate an SSL/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 Chapter 6, "Scenarios: Getting started with MQIPT," on page 69.
- 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 an SSL/TLS server scenario, with the addition of a certificate exit.

By changing the value of the **`SSLExitData`** property, the SSL/TLS connection between the two MQIPT servers can be allowed or rejected.

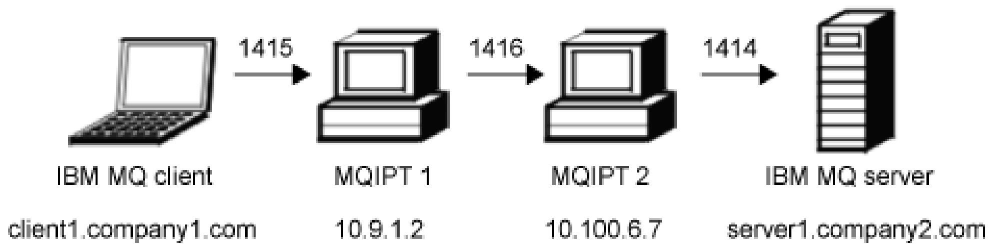


*Figure 23. SSL/TLS server network diagram*

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 SSL/TLS server, complete the following steps:

1. On MQIPT1:
   a. Open a command prompt and enter the following commands:
      ```
      C:
      cd \mqipt\exits
      javac -classpath C:\mqipt\lib\com.ibm.mq.ipt.jar;. SampleCertificateExit.java
      ```
   b. Edit `mqipt.conf` and add a route definition:
      ```
      [route]
      ListenerPort=1415
      Destination=9.100.6.7
      DestinationPort=1416
      SSLClient=true
      SSLClientKeyRing=C:\mqipt\ssl\sslSample.pfx
      SSLClientKeyRingPW=C:\mqipt\ssl\sslSample.pwd
      SSLClientExit=true
      SSLExitName=SampleCertificateExit
      SSLExitPath=C:\mqipt\exits
      SSLExitData=allow
      ```
   c. Open a command prompt and start MQIPT:
      ```
      C:\mqipt\bin\mqipt C:\mqiptHome
      ```

      where `C:\mqiptHome` indicates the location of the MQIPT configuration file, `mqipt.conf`.

      The following message indicates successful completion:
      ```
      5639-L92 (C) Copyright IBM Corp. 2000, 2017 All Rights Reserved
      MQCPI001 IBM MQ Internet
      Pass-Thru Version 2.1.0.3 starting
      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 protocols
      MQCPI036 ....SSL Client side enabled with properties :
      MQCPI031 ......CipherSuites <null>
      MQCPI032 ......keyring file C:\ssl\mqipt\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:\mqipt\exits\SampleCertificateExit
      MQCPI131 ......and certificate exit data 'allow'
      MQCPI078 Route 1415 ready for connection requests
      ```
2. On MQIPT2:
   a. Edit `mqipt.conf` and add a route definition:
      ```
      [route]
      ListenerPort=1416
      Destination=Server1.company2.com
      DestinationPort=1414
      SSLServer=true
      SSLServerKeyRing=C:\mqipt\ssl\sslSample.pfx
      SSLServerKeyRingPW=C:\mqipt\ssl\sslSample.pwd
      ```
   b. Open a command prompt and start MQIPT:
      ```
      C:
      cd \mqipt\bin
      mqipt ..
      ```

      (.. indicates that the MQIPT configuration file, `mqipt.conf`, is in the parent directory.)

The following message indicates successful completion:

```
5639-L92 (C) Copyright IBM Corp. 2000, 2017 All Rights Reserved
MQCPI001 IBM MQ Internet
Pass-Thru Version 2.1.0.3 starting
MQCPI011 The path C:\mqiptHome\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 protocols
MQCPI037 ....SSL Server side enabled with properties :
MQCPI031 ......CipherSuites <null>
MQCPI032 ......keyring file C:\mqipt\ssl\sslSample.pfx
MQCPI047 ......CA keyring 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, 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.

# Chapter 7. Administering and configuring MQIPT

Configure MQIPT by making changes to the configuration file `mqipt.conf`.

**Note:** You should set secure file permissions on the directory where `mqipt.conf` is located to prevent unauthorized users seeing the MQIPT password or changing its configuration.

You can edit the `mqipt.conf` configuration file either by using the Administration Client or by using a text editor of your choice and command line commands. Both techniques are described here, with reference information relevant to both:

- "Using the MQIPT Administration Client"
- "Administering MQIPT by using the command line" on page 118
- "Configuration reference information" on page 119
- "Making backups" on page 142
- "Performance tuning" on page 143

## Using the MQIPT Administration Client

You can use the Administration Client to configure and update one or more instances of MQIPT.

The Administration Client displays global properties and route-specific properties for each instance of MQIPT.

The only data stored locally by the Administration Client is the list of instances of MQIPT, in a file called `client.conf`. Global and route properties are always retrieved from MQIPT before they are displayed in the Administration Client. You therefore always see the current state of each instance of MQIPT.

### Starting the Administration Client

Start the Administration Client by using the `mqiptGui` script found in the MQIPT `bin` subdirectory.

The first time that the Administration Client is started, you are prompted for connection information to an instance of MQIPT. You must enter the following information:

**MQIPT Name**
A name of your choice that will be used to describe this instance of MQIPT.

**Network Address**
The address of the system where this instance of MQIPT is installed. The address can be a name recognized by the name server, a dotted decimal address, or localhost (if this instance is on the same computer as the Administration Client).

**Command Port**
The number of the port on which this instance of MQIPT is listening.

**Timeout (sec)**
The number of seconds that the Administration Client waits for a connection to this instance of MQIPT. Keep this value as low as possible to reduce the refresh time.

**Access Password**
The password used when communicating with this instance of MQIPT. Complete this field only if password checking is in force. (Password checking is in force if **`AccessPW`** is set in the global properties section of the `mqipt.conf` configuration file and it has a value of anything other than a null string.)

**Save Password**
Select the **Save Password** checkbox to save the password locally for future sessions. If **Save Password** is cleared, the password is saved only for the duration of the current session, or until this instance of MQIPT is removed from the Administration Client).

## Administering an instance of MQIPT

You can view and update the global and route properties of instances of MQIPT by using the Administration Client.

See "Starting the Administration Client" on page 115 to learn how to start the Administration Client.

Select an instance of MQIPT from the list to retrieve the global and route properties from this instance of MQIPT. If this instance of MQIPT is not running, or the correct value of **`CommandPort`** has not been specified in the global properties section of the `mqipt.conf` configuration file, an error message is issued. To change the host name and command port, click **MQIPT** > **Connection**.

Double-click an instance of MQIPT in the list to show a list of available routes. Select a route to change its properties.

You can add a route by clicking **MQIPT** > **Add Route**. The default property values for the new route, as defined by the global properties set for this instance of MQIPT, are shown.

Click **MQIPT** > **Apply** to apply the changes you have made. The changes are saved in the `mqipt.conf` configuration file of this instance of MQIPT and take effect immediately.

**Note:** Any comment lines that have been added in the `mqipt.conf` file are lost when it is updated.

## Setting MQIPT properties

The value used for each MQIPT property is determined by where it is set.

1. If you set the value of a parameter in the `[route]` section of the `mqipt.conf` configuration file, that value is used for the route and overides any value set in the `[global]` section. Route values are specific to a single route; they do not affect any other route.
2. If you set the value of a parameter in the `[global]` section of the `mqipt.conf` configuration file, that value is used for all routes unless explicitly overridden for a particular route.

3. All MQIPT properties have default values that are used if they are not explicitly set in either the [global] or [route] section of the mqipt.conf configuration file. See "Summary of properties" on page 120 for a list of these default values.

### Example

The following example shows how the values used for the two parameters **LDAPCacheTimeout** and **MinConnectionThreads** are affected by the place where parameters are set.

Settings in the mqipt.conf configuration file:

```
[global]
LDAPCacheTimeout = 120

[route]
name = route1
MinConnectionThreads = 10

[route]
name = route2
LDAPCacheTimeout = 60
```

Values used by MQIPT:

```
route1
   LDAPCacheTimeout:    120 (not set in route1 [route] section; set in [global] section)
   MinConnectionThreads: 10 (set in [route] section)
route2
   LDAPCacheTimeout:     60 (set in [route] section; [global] section ignored)
   MinConnectionThreads:  5 (not set in mqipt.conf for route2; uses the default value)
```

## Administration Client menu options

### File menu

You can manage the list of MQIPT instances by using the following options that are available on the **File** menu:

**Add MQIPT**
: Adds a new instance of MQIPT to the list in Administration Client. See "Starting the Administration Client" on page 115 for details of the information that you must enter.

**Remove MQIPT**
: Removes the currently highlighted instance of MQIPT from the list in Administration Client. This option does not stop or affect the running of this instance of MQIPT.

**Save Configuration**
: Saves the list of MQIPT instances to the local Administration Client configuration file so that they can be restored the next time that Administration Client starts. Only this MQIPT is saved locally; [global] and [route] properties are always retrieved from each instance of MQIPT.

**Quit**
: Stops Administration Client running. You are given the option to save outstanding changes before Administration Client closes.

### MQIPT menu

You can manage the selected instance of MQIPT by using the following options that are available on the **MQIPT** menu:

**Connection**
: Changes the access properties of an instance MQIPT. See "Starting the Administration Client" on page 115 for details of the information that you can update.

**Password**
: Changes the password required to access an instance of MQIPT. Leave the **Current Password** field blank if there is no password currently set. Do not enter a new password if you want to stop using a password. Select the **Save Password** check box if you want to save the password locally. If you do not save the password, you must enter it evey time you want to access this instance of MQIPT.

**Add Route**
: Adds a route to a selected instance of MQIPT. Each route must have a unique listener port for an instance of MQIPT.

**Delete Route**
: Deletes the selected route from the instance of MQIPT. The deletion does not take effect until it is applied, by clicking **MQIPT** > **Apply**.

**Apply**
: Updates the configuration file of an instance of MQIPT. The new settings are made effective immediately.

**Refresh**
: Reads the current configuration file from the selected instance of MQIPT and refreshes the display.

**Stop**
: Stops an instance of MQIPT from running. After this command, you lose contact with the MQIPT. This command is ignored unless the global property RemoteShutdown is turned on.

Route properties can be updated in the same way as MQIPT global properties. When you change any properties of a route, you must apply the changes to make them take effect. You can do this either by selecting the **MQIPT** > **Apply** menu option or replying `Yes` when you are prompted to save the configuration.

## Administering MQIPT by using the command line

If you choose not to use the Administration Client, you can use the command line to administer and configure MQIPT.

Using an editor of choice, change the configuration file, `mqipt.conf`, to meet your requirements. See "Configuration reference information" on page 119 for a list of the properties you can change.

If the `[global]` section of `mqipt.conf` specifies a value for `CommandPort`, MQIPT listens on this port for commands from the mqiptAdmin script:

```
mqiptAdmin -refresh {hostname {port} }       sends the refresh command
mqiptAdmin -stop    {hostname {port} }       sends the stop command
```

The `mqiptAdmin` script is in the `bin` subdirectory.

If you do not provided values for *hostname* and *port*, *hostname* defaults to `localhost` and *port* defaults to 1881.

**`refresh`**
MQIPT rereads `mqipt.conf` and takes the following actions:

- If any of the routes currently active are marked as inactive (or are no longer specified), MQIPT closes those routes and stops listening for incoming connections on them.
- Any routes marked active in the configuration file that it does not currently have running, it starts them up.
- if the configuration parameters of a currently running route have changed, MQIPT applies the changed values to those routes. Where possible (for example, a change to the setting of trace) it does this without disruption to running connections. For some parameter changes (for example, a change to a destination), MQIPT has to close all connections before effecting the change and restarting the route.

Using the MQIPT **Apply** menu option of the Administration Client has the same effect, provided that the Administration Client has not changed any of the MQIPT settings.

**`stop`**

MQIPT closes all connections, stops listening for incoming connections, and then exits.

This command is ignored unless the `mqipt.conf` file specifies `RemoteShutDown=true`.

Using the MQIPT **Stop** menu option of the Administration Client has the same effect.

**Note:** On Windows systems, these administrative functions are also available from the **Start** > **Programs** menu.

## Configuration reference information

MQIPT uses a configuration file called `mqipt.conf` to define routes and to control the actions of the MQIPT server.

The configuration file comprises a number of sections. There is one `[global]` section, and an additional `[route]` section for each route that has been defined through MQIPT.

Each section contains name/value property pairs. Some properties can appear only in the `[global]` section, some can appear only in the `[route]` sections, and some can appear both in `[route]` and `[global]` sections. If a property appears in both route and `[global]` sections, the value of the property in the `[route]` section overrides the global value, but only for the route in question. In this way, the `[global]` section can be used to establish the default values to be used for those properties not set in the individual `[route]` sections.

The `[global]` section starts with a line containing the characters `[global]` and ends when the first `[route]` section starts. The `[global]` section must precede all `[route]` sections in the file.

Each [route] section starts with a line containing the characters [route] and ends when the next [route] section starts, or when the end of the configuration file is reached.

Any unrecognized property name is ignored. If a property in a [route] section has a recognized name but has an invalid value (for example MinConnectionThreads=x or HTTP=unsure), that route is disabled (that is, it does not listen for any incoming connections). If a property in the [global] section has a recognized name but has an invalid value, all routes are disabled and MQIPT does not start. Where a property is listed as taking the values true or false, any mixture of uppercase and lowercase can be used.

You can change the value of a property by either editing the mqipt.conf file or by using the Administration Client GUI. To apply any changes, refresh MQIPT, either from the Administration Client GUI or by using the mqiptAdmin -refresh command.

Changes to certain properties cause a route to be restarted only if other properties are already enabled. For example, any changes to the HTTP properties have an effect only if the **HTTP** property is also enabled.

When a route is restarted, existing connections are terminated. To override this behavior, set the **RouteRestart** property to false. This prevents the route from restarting, allowing existing connections to remain active until the **RouteRestart** property is reenabled.

For information about how to set up some simple configurations, see Chapter 6, "Scenarios: Getting started with MQIPT," on page 69. For a sample configuration, see the mqiptSample.conf file in the MQIPT installation directory.

## Summary of properties

This table shows a summary of MQIPT configuration properties and includes the following information:

- An alphabetical list of MQIPT properties with links to further information in the [route] section, or the [global] section if the [route] section does not apply.
- The property that must be set to true for a value to have an effect.
- Whether the property applies to the [global] section, the [route] section, or both.
- Default values that are used if a property is missing from both the [route] section and the [global] section. When specifying the values true and false, any mixture of uppercase and lowercase characters can be used.

| Name of property | Property to set true | Global | Route | Default |
|---|---|---|---|---|
| AccessPW | | yes | no | null |
| Active | | yes | yes | true |
| ClientAccess | | yes | yes | false |
| CommandPort | | yes | no | null |
| ConnectionLog | | yes | no | true |
| Destination | | no | yes | null |
| DestinationPort | | no | yes | 1414 |

| Name of property | Property to set `true` | Global | Route | Default |
|---|---|---|---|---|
| HTTP | | yes | yes | `false` |
| HTTPProxy | HTTP | yes | yes | null |
| HTTPProxyPort | HTTP | yes | yes | `8080` |
| HTTPS | HTTP | yes | yes | `false` |
| HTTPServer | HTTP | yes | yes | null |
| HTTPServerPort | HTTP | yes | yes | null |
| IdleTimeout | | yes | yes | `0` |
| IgnoreExpiredCRLs | | yes | yes | `false` |
| LDAP | | yes | yes | `false` |
| LDAPIgnoreErrors | LDAP | yes | yes | `false` |
| LDAPCacheTimeout | LDAP | yes | yes | `24` |
| LDAPSaveCRLs (Note 3) | LDAP | yes | yes | `false` |
| LDAPServer1 | LDAP | yes | yes | null |
| LDAPServer1Port | LDAP | yes | yes | `389` |
| LDAPServer1Userid | LDAP | yes | yes | null |
| LDAPServer1Password | LDAP | yes | yes | null |
| LDAPServer1Timeout | LDAP | yes | yes | `0` |
| LDAPServer2 | LDAP | yes | yes | null |
| LDAPServer2Port | LDAP | yes | yes | `389` |
| LDAPServer2Userid | LDAP | yes | yes | null |
| LDAPServer2Password | LDAP | yes | yes | null |
| LDAPServer2Timeout | LDAP | yes | yes | `0` |
| ListenerAddress | | yes | yes | null |
| ListenerPort | | no | yes | null |
| LocalAddress | | yes | yes | null |
| MaxConnectionThreads | | yes | yes | `100` |
| MaxLogFileSize | | yes | no | `50` |
| MinConnectionThreads | | yes | yes | `5` |
| Name | | no | yes | null |
| NDAdvisor | | yes | yes | `false` |
| NDAdvisorReplaceMode | NDAdvisor | yes | yes | `false` |
| OutgoingPort | | no | yes | `0` |
| QMgrAccess | | yes | yes | `true` |
| RemoteShutdown | | yes | no | `false` |
| RouteRestart | | yes | yes | `true` |
| SecurityExit | | yes | yes | `false` |
| SecurityExitName | SecurityExit | yes | yes | null |
| SecurityExitPath | SecurityExit | yes | yes | *`mqipt_home`*`\exits` |
| SecurityExitTimeout | SecurityExit | yes | yes | `30` |

| **Name of property** | **Property to set `true`** | **Global** | **Route** | **Default** |
|---|---|---|---|---|
| SecurityManager | | yes | no | `false` |
| SecurityManagerPolicy | | yes | no | null |
| SocksClient | | yes | yes | `false` |
| SocksProxyHost | SocksClient | yes | yes | null |
| SocksProxyPort | SocksClient | yes | yes | `1080` |
| SocksServer | | yes | yes | `false` |
| SSLClient | | yes | yes | `false` |
| SSLClientCAKeyRing | SSLClient | yes | yes | null |
| SSLClientCAKeyRingPW | SSLClient | yes | yes | null |
| SSLClientCipherSuites | SSLClient | yes | yes | null |
| SSLClientConnectTimeout | SSLClient | yes | yes | `30` |
| SSLClientDN_C | SSLClient | yes | yes | * (Note 1) |
| SSLClientDN_CN | SSLClient | yes | yes | * (Note 1) |
| SSLClientDN_DC | SSLClient | yes | yes | * (Note 1) |
| SSLClientDN_DNQ | SSLClient | yes | yes | * (Note 1) |
| SSLClientDN_L | SSLClient | yes | yes | * (Note 1) |
| SSLClientDN_O | SSLClient | yes | yes | * (Note 1) |
| SSLClientDN_OU | SSLClient | yes | yes | * (Note 1) |
| SSLClientDN_PC | SSLClient | yes | yes | * (Note 1) |
| SSLClientDN_ST | SSLClient | yes | yes | * (Note 1) |
| SSLClientDN_Street | SSLClient | yes | yes | * (Note 1) |
| SSLClientDN_T | SSLClient | yes | yes | * (Note 1) |
| SSLClientDN_UID | SSLClient | yes | yes | * (Note 1) |
| SSLClientExit | | yes | yes | `false` |
| SSLClientKeyRing | SSLClient | yes | yes | null |
| SSLClientKeyRingPW | SSLClient | yes | yes | null |
| SSLClientProtocols | SSLClient | yes | yes | SSLv3, TLSv1, TLSv1.1, TLSv1.2 |
| SSLClientSiteDN_C | SSLClient | yes | yes | * (Note 1) |
| SSLClientSiteDN_CN | SSLClient | yes | yes | * (Note 1) |
| SSLClientSiteDN_DC | SSLClient | yes | yes | * (Note 1) |
| SSLClientSiteDN_DNQ | SSLClient | yes | yes | * (Note 1) |
| SSLClientSiteDN_L | SSLClient | yes | yes | * (Note 1) |
| SSLClientSiteDN_O | SSLClient | yes | yes | * (Note 1) |
| SSLClientSiteDN_OU | SSLClient | yes | yes | * (Note 1) |
| SSLClientSiteDN_PC | SSLClient | yes | yes | * (Note 1) |
| SSLClientSiteDN_ST | SSLClient | yes | yes | * (Note 1) |
| SSLClientSiteDN_Street | SSLClient | yes | yes | * (Note 1) |
| SSLClientSiteDN_T | SSLClient | yes | yes | * (Note 1) |

| Name of property | Property to set **true** | Global | Route | Default |
|---|---|---|---|---|
| SSLClientSiteDN_UID | SSLClient | yes | yes | * (Note 1) |
| SSLClientSiteLabel | SSLClient | yes | yes | null |
| SSLExitData | SSLServerExit | yes | yes | null |
| SSLExitName | SSLServerExit | yes | yes | null |
| SSLExitPath | SSLServerExit | yes | yes | *mqipt_home*\ exits |
| SSLExitTimeout | SSLServerExit | yes | yes | 30 |
| SSLProxyMode | | yes | yes | false |
| SSLPlainConnections | either SSLServer or SSLProxyMode | yes | yes | false |
| SSLServer | | yes | yes | false |
| SSLServerAskClientAuh | SSLServer | yes | yes | false |
| SSLServerCAKeyRing | SSLServer | yes | yes | null |
| SSLServerCAKeyRingPW | SSLServer | yes | yes | null |
| SSLServerCipherSuites | SSLServer | yes | yes | null |
| SSLServerDN_C | SSLServer | yes | yes | * (Note 1) |
| SSLServerDN_CN | SSLServer | yes | yes | * (Note 1) |
| SSLServerDN_DC | SSLServer | yes | yes | * (Note 1) |
| SSLServerDN_DNQ | SSLServer | yes | yes | * (Note 1) |
| SSLServerDN_L | SSLServer | yes | yes | * (Note 1) |
| SSLServerDN_O | SSLServer | yes | yes | * (Note 1) |
| SSLServerDN_OU | SSLServer | yes | yes | * (Note 1) |
| SSLServerDN_PC | SSLServer | yes | yes | * (Note 1) |
| SSLServerDN_ST | SSLServer | yes | yes | * (Note 1) |
| SSLServerDN_Street | SSLServer | yes | yes | * (Note 1) |
| SSLServerDN_T | SSLServer | yes | yes | * (Note 1) |
| SSLServerDN_UID | SSLServer | yes | yes | * (Note 1) |
| SSLServerExit | | yes | yes | false |
| SSLServerKeyRing | SSLServer | yes | yes | null |
| SSLServerKeyRingPW | SSLServer | yes | yes | null |
| SSLServerProtocols | SSLServer | yes | yes | SSLv3, TLSv1, TLSv1.1, TLSv1.2 |
| SSLServerSiteDN_C | SSLServer | yes | yes | * (Note 1) |
| SSLServerSiteDN_CN | SSLServer | yes | yes | * (Note 1) |
| SSLServerSiteDN_DC | SSLServer | yes | yes | * (Note 1) |
| SSLServerSiteDN_DNQ | SSLServer | yes | yes | * (Note 1) |
| SSLServerSiteDN_L | SSLServer | yes | yes | * (Note 1) |
| SSLServerSiteDN_O | SSLServer | yes | yes | * (Note 1) |
| SSLServerSiteDN_OU | SSLServer | yes | yes | * (Note 1) |

| Name of property | Property to set `true` | Global | Route | Default |
|---|---|---|---|---|
| SSLServerSiteDN_PC | SSLServer | yes | yes | * (Note 1) |
| SSLServerSiteDN_ST | SSLServer | yes | yes | * (Note 1) |
| SSLServerSiteDN_Street | SSLServer | yes | yes | * (Note 1) |
| SSLServerSiteDN_T | SSLServer | yes | yes | * (Note 1) |
| SSLServerSiteDN_UID | SSLServer | yes | yes | * (Note 1) |
| SSLServerSiteLabel | SSLServer | yes | yes | null |
| TCPKeepAlive | | yes | yes | false |
| Trace | | yes | yes | `0` |
| UriName | HTTP | yes | yes | (Note 2) |

**Notes:**

1. The asterisk (*) represents a wildcard.
2. See UriName in "Route properties" on page 125 for details about the default settings.
3. **`LDAPSaveCRLs`** is deprecated in version 2.1 of MQIPT. The parameter is included for compatibility with earlier versions, but has no effect.

**Related information**:

"Configuration reference information" on page 119
MQIPT uses a configuration file called `mqipt.conf` to define routes and to control the actions of the MQIPT server.

"Global properties"
The `mqipt.conf` configuration file can contain a number of global properties.

"Route properties" on page 125
The `mqipt.conf` configuration file can contain properties for individual routes.

## Global properties

The `mqipt.conf` configuration file can contain a number of global properties.

The following properties can appear only in the `[global]` section of `mqipt.conf`. All the route properties except **`ListenerPort`**, **`Destination`**, **`DestinationPort`**, **`Name`**, and **`OutgoingPort`** can also appear in the `[global]` section. If a property appears in both route and `[global]` sections, the value of the property in the `[route]` section overrides the global value, but only for the route in question. In this way, the `[global]` section can be used to establish the default values to be used for those properties not set in the individual `[route]` sections.

**`AccessPW`**
The password used when an Administration Client sends commands to the MQIPT. If this property is not present or is set to blank, no checking takes place.

**`CommandPort`**
The TCP/IP port on which MQIPT listens for configuration commands from the **`mqiptAdmin`** utility or the Administration Client. You can change the command port from the Administration Client in the same way as any other property. Note that you do not change the connection properties. When you apply the new setup to the MQIPT, the Administration Client changes the connection properties automatically.

If the **CommandPort** property is not present, MQIPT does not listen for configuration commands. To use the default port number, 1881, used by default by both the Administration Client and by the **mqiptAdmin** script from the command line, set **CommandPort** to 1881. This value is set for you if you use the `mqiptSample.conf` configuration file.

**ConnectionLog**
Either `true` or `false`. When true, MQIPT logs all connection attempts (successful or otherwise) in the `logs` subdirectory and disconnection events to the file `mqipt`*YYYYMMDDHHmmSS*`.log` (where *YYYYMMDDHHmmSS* are characters representing the current date and time). The default value of **ConnectionLog** is `true`. When this property is changed from `true` to `false`, MQIPT closes the existing connection log and creates a new one. The new log is used when the property is reset to `true`.

**MaxLogFileSize**
The maximum size (specified in KB) of the connection log file. When the file size increases above this maximum a backup copy (`mqipt001.log`) is made, and a new file is started. Only two backup files are kept (`mqipt001.log` and `mqipt002.log`); each time the main log file fills up, any earlier backups are erased. The default value of **MaxLogFileSize** is `50`; the minimum allowed value is `5`.

**RemoteShutDown**
Either `true` or `false`. When true (and when there is a command port) MQIPT shuts down whenever a stop command is received on the command port. The default value is `false`.

**SecurityManager**
Set this property to `true` to enable the Java Security Manager for this instance of MQIPT. You must ensure that the correct permissions are granted. See "Java Security Manager" on page 49 for more information. The default value for this property is `false`.

**SecurityManagerPolicy**
The fully-qualified file name of a policy file. If this property is not set then only the default system and user policy files are used. If the Java Security Manager is already enabled, then changes to this property have no effect until the Java Security Manager has been disabled and reenabled.

**Trace**

The level of trace for global MQIPT threads that are not associated with a route, and for routes that have no **Trace** property set. For example, the main MQIPT control thread and the command port listener thread are not associated with a route and are only traced if trace is enabled in the `[global]` section. The value of the **Trace** property in a `[route]` section overrides the global **Trace** property, for that route. For information about tracing threads associated with a route, see **Trace** in the `[route]` section.

This property should be an integer in the range `0 - 5`, where `0` indicates that trace is disabled, and any other value indicates that trace is enabled. The default value is `0`.

## Route properties

The `mqipt.conf` configuration file can contain properties for individual routes.

The `[route]` section of the `mqipt.conf` configuration file can contain the following properties:

**Active**
The route accepts incoming connections only if the value of **Active** is set to `true`. This means that you can temporarily shut off access to the destination, by setting this value to `false`, without having to delete the `[route]` section from the configuration file. If you change this property to `false`, the route is stopped when a refresh command is issued. All connections to the route are stopped.

**ClientAccess**
The route allows incoming client channel connections only if the value of **ClientAccess** is set to `true`. Note that potentially you can configure MQIPT to accept client requests only, queue manager requests only, or both types of request. Use this property in conjunction with the **QMgrAccess** property. If you change this property to `false`, the route is stopped, and restarted when a refresh command is issued. All connections to the route are stopped.

**Destination**
The host name (or dotted decimal IP address) of the queue manager (or subsequent MQIPT object) to which this route is to connect. Each `[route]` section must contain an explicit **Destination** value, but several `[route]` sections can refer to the same destination. If a change to this property affects a route, the route is stopped, and restarted when a refresh command is issued. All connections to the route are stopped. When using the **SocksProxyHost** property the **Destination** property must use the dotted decimal format.

**DestinationPort**
The port on the destination host to which this route is to connect. Each `[route]` section must contain an explicit **DestinationPort** value, but several routes can refer to the same combination of **Destination** and **DestinationPort**values. If a change to this property affects a route, the route is stopped, and restarted when a refresh command is issued. All connections to the route are stopped.

**HTTP**
Set **HTTP** to `true` for routes responsible for making outbound HTTP tunneling requests. The **Destination** property for the route must be the host name of another MQIPT when HTTP it set to true. Set **HTTP** to `false` for routes connected to IBM MQ queue managers. If you change this property, the route is stopped. At least one of the **HTTPProxy** or **HTTPServer** properties must also be specified when HTTP is set to true. This property cannot be used in conjunction with the **SocksClient** property.

**HTTPProxy**
The host name (or dotted decimal IP address) of the HTTP proxy used by all connections for this route. If **HTTPServer** is also defined, then a **CONNECT** request is issued to the HTTP proxy, instead of a normal **POST** request. If you change this property (and **HTTP** is set to `true`), the route is stopped, and restarted when a refresh command is issued. All connections to the route are stopped.

**HTTPProxyPort**
The port address to use on the HTTP proxy. The default value is `8080` unless **HTTPS** has been set to `true` and **HTTPServer** is not set, in which case the default is `443`. If you change this property (and **HTTP** is set to `true`), the route is stopped, and restarted when a refresh command is issued. All connections to the route are stopped.

**HTTPServer**
The host name (or dotted decimal IP address) of the HTTP server used by all connections for this route. This can be either the host name of another MQIPT, or a HTTP proxy.

If **HTTPProxy** is not specified, MQIPT connects to the host specified in **HTTPServer**, and issues HTTP POST requests to the host specified in the route **Destination** property. If **HTTPProxy** is specified, MQIPT connects to the host specified in **HTTPProxy** instead.

If you change this property (and **HTTP** is set to true), the route is stopped, and restarted when a refresh command is issued. All connections to the route are stopped.

**HTTPS**
: Set **HTTPS** to true to make HTTPS requests. The **HTTP** and **SSLClient** properties must also be enabled, and the **SSLClientKeyRing** and **SSLClientKeyRingPW** properties set as for SSL/TLS operation. If you change the **HTTPS** property (and **HTTP** is set to true), the route is stopped, and restarted when a refresh command is issued. All connections to the route are stopped.

**HTTPServerPort**
: The port address to use on the HTTP server. The default value is 8080, unless **HTTPS** has been set to true, in which case the default is 443. If you change this property (and **HTTP** is set to true), the route is stopped, and restarted when a refresh command is issued. All connections to the route are stopped.

**IdleTimeout**
: The time, in minutes, after which an idle connection is closed. Note that queue manager to queue manager channels also have the **DISCINT** property. If you set the **IdleTimeout** parameter, take note of **DISCINT**. If **IdleTimeout** is set to 0, there is no idle timeout. Changes to this property take effect only when the route is restarted.

**IgnoreExpiredCRLs**
: Set **IgnoreExpiredCRLs** to true to ignore an expired CRL. The default value is false. Note that if you set **IgnoreExpiredCRLs** to true, a revoked certificate could be used to make an SSL/TLS connection.

**LDAP**
: Set **LDAP** to true to enable use of an LDAP server when using SSL/TLS connections. MQIPT will use the LDAP server to retrieve CRLs and ARLs. The **SSLClient** property or **SSLServer** property must also be set to true for this property to take effect.

**LDAPCacheTimeout**
: The expiry time, in hours, of the temporary cache in which a CRL retrieved from an LDAP server, is stored. After this time, the entire CRL cache is emptied. For example, specifying a value of 1 hour means that the cache is emptied once per hour. The default value is 24. If you specify a timeout value of 0, entries in the cache will not expire until the route is restarted. If you change this property (and **LDAP** is set to true), the route is stopped, and restarted when a refresh command is issued. All connections to the route are stopped.

**LDAPIgnoreErrors**
: Set **LDAPIgnoreErrors** to true to ignore any connection or timeout errors when performing an LDAP search. If MQIPT cannot perform a successful search, it will not allow the client connection to complete, unless this property has been enabled. A successful search means that a CRL has been retrieved or there are no CRLs available for the specified CA. If you change this property (and **LDAP** is set to true), the route is stopped, and restarted when a refresh command is issued. All connections to the route are stopped.

**Note:** If you enable this property, a revoked certificate could be used to make an SSL/TLS connection.

**LDAPSaveCRLs**
**LDAPSaveCRLs** is deprecated in version 2.1 of MQIPT. The parameter is included for compatibility with earlier versions, but has no effect. LDAP CRLs are therefore no longer cached on disk in the key-ring file, but are retrieved after each restart.

**LDAPServer1**
The host name or IP address of the main LDAP server. This property must be set if LDAP has been set to `true`. If you change this property (and **LDAP** is set to `true`), the route is stopped, and restarted when a refresh command is issued. All connections to the route are stopped.

**LDAPServer1Port**
The listening port number of the main LDAP server. The default value is `389`. If you change this property (and **LDAP** is set to `true`), the route is stopped, and restarted when a refresh command is issued. All connections to the route are stopped.

**LDAPServer1Userid**
The user ID needed to access the main LDAP server. This property must be set if authorization to access the main LDAP server is required. If you change this property (and **LDAP** is set to `true`), the route is stopped, and restarted when a refresh command is issued. All connections to the route are stopped.

**LDAPServer1Password**
The password needed to access the main LDAP server. This property must be set if **LDAPServer1Userid** has been set to `true`. If you change this property (and **LDAP** is set to `true`), the route is stopped, and restarted when a refresh command is issued. All connections to the route are stopped.

**LDAPServer1Timeout**
The time, in seconds, that MQIPT waits for a response from the main LDAP server. The default value is `0`, which means the connection will not time out. If you change this property (and **LDAP** is set to `true`), the route is stopped, and restarted when a refresh command is issued. All connections to the route are stopped.

**LDAPServer2**
The host name or IP address of the backup LDAP server. This property is optional. If you change this property (and **LDAP** is set to `true`), the route is stopped, and restarted when a refresh command is issued. All connections to the route are stopped.

**LDAPServer2Port**
The listening port number of the backup LDAP server. The default value is `389`. If you change this property (and **LDAP** is set to `true`), the route is stopped, and restarted when a refresh command is issued. All connections to the route are stopped.

**LDAPServer2Userid**
The userid needed to access the backup LDAP server. This property must be set if authorization to access the backup LDAP server is required. If you change this property (and **LDAP** is set to `true`), the route is stopped, and restarted when a refresh command is issued. All connections to the route are stopped.

**LDAPServer2Password**
The password needed to access the backup LDAP server. This property must

be set if **LDAPServer2** has been set to `true`. If you change this property (and **LDAP** is set to `true`), the route is stopped, and restarted when a refresh command is issued. All connections to this route are stopped.

**LDAPServer2Timeout**
The time, in seconds, that MQIPT will wait for a response from the backup LDAP server. The default value is `0`, which means the connection will not time out. If you change this property (and **LDAP** is set to `true`), the route is stopped, and restarted when a refresh command is issued. All connections to the route are stopped.

**ListenerAddress**
Use this property if the MQIPT system has multiple IP addresses and you need to bind the route listener port to a specific address. This is useful for restricting inbound connections to those from a particular network interface. The value of this property should be an IP address belonging to one of the network interfaces on the system where MQIPT is running. The default is to accept connections from all network interfaces.

**ListenerPort**
The port number on which the route should listen for incoming requests. Each `[route]` section must contain an explicit **ListenerPort** value The **ListenerPort** values set in each section must be distinct. Any valid port number can be used, including ports `80` and `443`, provided that the ports chosen are not already in use by any other TCP/IP listener running on the same host.

**LocalAddress**
The IP address to bind all connections to for this route on this computer. The chosen address must be an IP address that is associated with one of the network interfaces on the computer on which MQIPT is running. If you change this property, the route is stopped, and restarted when a refresh command is issued. All connections to the route are stopped.

**MaxConnectionThreads**
The maximum number of connection threads, and thus the maximum number of concurrent connections, that can be handled by this route. If this limit is reached, the **MaxConnectionThreads** value also indicates the number of connections that are queued when all the threads are in use. Beyond that number, subsequent connection requests are refused. The minimum allowed value is the greater of `1` and the value of **MinConnectionThreads**. If a change to this property affects a route, the new value is used when the refresh command is issued. All connections use the new value immediately. The route is not stopped.

**MinConnectionThreads**
The number of connection threads allocated to handle incoming connections on a route when the route is started. The number of threads allocated does not drop below this value during the time the route is active. The minimum allowed value is the lesser of `0` and the value of **MaxConnectionThreads**. Changes to this property take effect only when the route is restarted.

**Name**
A name to help identify the route. This property is optional. The value is shown in console messages and tracing information. Changes to this property take effect only when the route is restarted.

**NDAdvisor**
Set **NDAdvisor** to `true` for routes managed by the Network Dispatcher to allow the route to respond to requests from the custom advisor. If you change this

property to `false`, the route is stopped when a refresh command is issued. All connections to the route are stopped. To use the **NDAdvisorReplaceMode** property, set **NDAdvisor** to `true`.

**NDAdvisorReplaceMode**
: Set **NDAdvisorReplaceMode** to `true` to use the replace mode of the Network Dispatcher custom advisor. You must have started the mqipt_replace custom advisor for the port number specified in **ListenerPort**. Set this property to `false` to use normal mode. You must set the **NDAdvisor** property to `true` to use this property.

**OutgoingPort**
: The starting port number used by outgoing connections. The range of port numbers match the **MaxConnectionThread** value for this route. The default value of `0` uses a system-defined port number. If you change this property, the route is stopped and restarted when a refresh command is issued. All connections to this route are stopped. When HTTP is used, each channel connection requires two outgoing ports. See "Port number control" on page 57.

**QMgrAccess**
: Set **QMgrAccess** to `true` to allow incoming queue manager channel connections (for example sender channels). If you change this property to `false`, the route is stopped when a refresh command is issued. All connections to this route are stopped.

**RouteRestart**
: Set **RouteRestart** to `false` to stop the route from restarting when other route properties have been changed and a refresh command has been issued. The default value for this property is `true`.

**SecurityExit**
: Set **SecurityExit** to `true` to enable a user-defined security exit. The default value for this property is `false`.

**SecurityExitName**
: The class name of the user-defined security exit. This property must be set if **SecurityExit** has been set to `true`. If you change this property (and **SecurityExit** is set to `true`), the route is stopped, and restarted when a refresh command is issued. All connections to this route are stopped.

**SecurityExitPath**
: The fully-qualified path name containing the user-defined security exit. If this property has not been set, then it will default to the exits subdirectory. This property can also define the name of a Java archive (JAR) file containing the user-defined security exit. If you change this property (and **SecurityExit** is set to `true`), the route is stopped and restarted when a refresh command is issued. All connections to this route are stopped.

**SecurityExitTimeout**
: The timeout value (in seconds) used by MQIPT to determine how long to wait for a response when validating a connection request. The default value is `30`. If you change this property (and **SecurityExit** is set to `true`), the route is stopped, and restarted when a refresh command is issued. All connections to the route are stopped.

**SocksClient**
: Set **SocksClient** to `true` to make the route act as a SOCKS client and define all connections through the SOCKS proxy with the **SocksProxyHost** and **SocksProxyPort** properties. If you change this property, the route is stopped,

and restarted when a refresh command is issued. All connections to the route are stopped. This property cannot be used with:

- **HTTP**
- **SocksServer**
- **SSLClient**
- **SSLProxyMode**

**SocksProxyHost**
The host name (or dotted decimal IP address) of the SOCKS proxy that all connections for this route use. If you change this property (and **SocksClient** is set to `true`), the route is stopped, and restarted when a refresh command is issued. All connections to this route are stopped. When using the **SocksProxyHost** property the **Destination** property must use the dotted decimal format.

**SocksProxyPort**
The port number to use on a SOCKS proxy. The default value is `1080`. If you change this property (and **SocksClient** is set to `true`), the route is stopped, and restarted when a refresh command is issued. All connections to the route are stopped.

**SocksServer**
Set **SocksServer** to `true` to make the route act as a SOCKS proxy and accept SOCKS client connections. If you change this property, the route is stopped, and restarted when a refresh command is issued. All connections to the route are stopped. This property cannot be used with the following properties:

- **SocksClient**
- **SSLProxyMode**
- **SSLServer**

**SSLClient**
Set **SSLClient** to `true` to make the route act as an SSL/TLS client and make outgoing SSL/TLS connections. Setting **SSLClient** to `true` implies that the destination is either another instance of MQIPT acting as an SSL/TLS server, or an HTTP proxy/server. You must specify the name of a key-ring file either with the **SSLClientKeyRing** property or the**SSLClientCAKeyRing** property. If you change **SSLClient**, the route is stopped, and restarted when a refresh command is issued. All connections to this route are stopped. This property cannot be used in conjunction with the following property:

- **SSLProxyMode**

**SSLClientCAKeyRing**
The fully-qualified file name of the key-ring file containing CA certificates, used to authenticate certificates from the SSL/TLS server. On Windows platforms, you must use a double backslash (\\) as the file separator. If you change this property (and **SSLClient** is set to `true`), the route is stopped, and restarted when a refresh command is issued. All connections to the route are stopped.

**SSLClientCAKeyRingPW**
The fully-qualified file name of the file containing the password to open the client CA key-ring file. On Windows platforms, you must use a double backslash (\\) as the file separator. If you change this property (and **SSLClient** is set to `true`), the route is stopped, and restarted when a refresh command is issued. All connections to this route are stopped.

**SSLClientCipherSuites**
The name of the SSL/TLS CipherSuite to use on the SSL/TLS client side. This can be one or more of the supported CipherSuites. If you leave this property blank, the SSL/TLS client uses the supported CipherSuites from the **SSLClientKeyRing**. If you change this property (and **SSLClient** is set to `true`), the route is stopped, and restarted when a refresh command is issued. All connections to this route are stopped.

**SSLClientConnectTimeout**
The time (in seconds) that an SSL/TLS client waits for an SSL/TLS connection to be accepted. If you change this property (and **SSLClient** is set to `true`), the route is stopped, and restarted when a refresh command is issued. All connections to the route are stopped.

**SSLClientDN_C**
Use this property to accept certificates received from the SSL/TLS server that match this country name. The name can be prefixed or suffixed with an asterisk (*) to extend its scope. Certificate matching is not case sensitive. If you do not specify this property, all country names are accepted. If you change this property (and **SSLClient** is set to `true`), the route is stopped, and restarted when a refresh command is issued. All connections to the route are stopped.

**SSLClientDN_CN**
Use this property to accept certificates received from the SSL/TLS server that match this common name. The name can be prefixed or suffixed with an asterisk (*) to extend its scope. Certificate matching is not case sensitive. If you do not specify this property, all common names are accepted. If you change this property (and **SSLClient** is set to `true`), the route is stopped, and restarted when a refresh command is issued. All connections to the route are stopped.

**SSLClientDN_DC**
Use this property to accept certificates received from the SSL/TLS server that match this domain component. The name can be prefixed or suffixed with an asterisk (*) to extend its scope. Certificate matching is not case sensitive. You can specify multiple DCs by separating them with commas. Each DC represents an element in a domain name, for example the domain name `example.ibm.com` is represented as `example,ibm,com` using commas to separate the multiple values. If you do not specify this property, all domain components are accepted. If you change this property (and **SSLClient** is set to `true`), the route is stopped, and restarted when a refresh command is issued. All connections to the route are stopped.

**SSLClientDN_DNQ**
Use this property to accept certificates received from the SSL/TLS server that match this domain qualifier. The name can be prefixed or suffixed with an asterisk (*) to extend its scope. Certificate matching is not case sensitive. If you do not specify this property, all domain qualifiers are accepted. If you change this property (and **SSLClient** is set to `true`), the route is stopped, and restarted when a refresh command is issued. All connections to the route are stopped.

**SSLClientDN_L**
Use this property to accept certificates received from the SSL/TLS server that match this location. The name can be prefixed or suffixed with an asterisk (*) to extend its scope. Certificate matching is not case sensitive. If you do not specify this property, you imply "all locations". If you change this property (and **SSLClient** is set to `true`), the route is stopped, and restarted when a refresh command is issued. All connections to the route are stopped.

**SSLClientDN_O**
: Use this property to accept certificates received from the SSL/TLS server that match this organization. The name can be prefixed or suffixed with an asterisk (*) to extend its scope. Certificate matching is not case sensitive. If you do not specify this property, certificates are accepted from all organizations. If you change this property (and **SSLClient** is set to `true`), the route is stopped, and restarted when a refresh command is issued. All connections to the route are stopped.

**SSLClientDN_OU**
: Use this property to accept certificates received from the SSL/TLS server that match this Organizational Unit (OU). The name can be prefixed or suffixed with an asterisk (*) to extend its scope. You can specify multiple OUs by separating them with commas. (Match a literal comma by prefixing it with a backslash (\) character.) Certificate matching is not case sensitive. If you do not specify this property, certificates are accepted with any OU name. If you change this property (and **SSLClient** is set to `true`), the route is stopped and restarted when a refresh command is issued. All connections to this route are stopped.

**SSLClientDN_PC**
: Use this property to accept certificates received from the SSL/TLS server that match this postal code. The name can be prefixed or suffixed with an asterisk (*) to extend its scope. Certificate matching is not case sensitive. If you do not specify this property, all postal codes are accepted. If you change this property (and **SSLClient** is set to `true`), the route is stopped, and restarted when a refresh command is issued. All connections to the route are stopped.

**SSLClientDN_ST**
: Use this property to accept certificates received from the SSL/TLS server that match this state. The name can be prefixed or suffixed with an asterisk (*) to extend its scope. Certificate matching is not case sensitive. If you do not specify this property, certificates are accepted from servers in all states. If you change this property (and **SSLClient** is set to `true`), the route is stopped, and restarted when a refresh command is issued. All connections to the route are stopped.

**SSLClientDN_Street**
: Use this property to accept certificates received from the SSL/TLS server that match this street name. The name can be prefixed or suffixed with an asterisk (*) to extend its scope. Certificate matching is not case sensitive. If you do not specify this property, all street names are accepted. If you change this property (and **SSLClient** is set to `true`), the route is stopped, and restarted when a refresh command is issued. All connections to the route are stopped.

**SSLClientDN_T**
: Use this property to accept certificates received from the SSL/TLS server that match this title. The name can be prefixed or suffixed with an asterisk (*) to extend its scope. Certificate matching is not case sensitive. If you do not specify this property, all titles are accepted. If you change this property (and **SSLClient** is set to `true`), the route is stopped, and restarted when a refresh command is issued. All connections to the route are stopped.

**SSLClientDN_UID**
: Use this property to accept certificates received from the SSL/TLS server that match this user ID. The name can be prefixed or suffixed with an asterisk (*) to extend its scope. Certificate matching is not case sensitive. If you do not specify this property, all user IDs are accepted. If you change this property

(and **SSLClient** is set to true), the route is stopped, and restarted when a refresh command is issued. All connections to the route are stopped.

**SSLClientExit**
Use this property to enable or disable the use of an exit when the route is acting as an SSL/TLS client. This allows you to define exit details in the configuration file without them actually being used.

**SSLClientKeyRing**
The fully-qualified file name of the key-ring file containing the client certificate; on Windows platforms, you must use a double backslash (\\) as the file separator. You must specify **SSLClientKeyRing** if you set **SSLClient** to true. If you change **SSLClientKeyRing** (and **SSLClient** is set to true), the route is stopped, and restarted when a refresh command is issued. All connections to the route are stopped.

**SSLClientKeyRingPW**
The fully-qualified file name containing the password to open the client key-ring file; on Windows platforms, you must use a double backslash (\\) as the file separator. You must specify **SSLClientKeyRingPW** if you set **SSLClient** to true. If you change **SSLClientKeyRingPW** (and **SSLClient** is set to true), the route is stopped, and restarted when a refresh command is issued. All connections to the route are stopped.

**SSLClientProtocols**
Used to restrict the set of enabled secure socket protocols that are used to make outbound connections to the destination for a route when **SSLClient** is set to true.

You can specify multiple values by separating them with commas. If you do not specify this property, all supported JSSE protocols are enabled by default with the exception of SSL 3.0. From version 2.1.0.2 of MQIPT, you must explicitly set SSLv3 on this property for SSL 3.0 to be enabled.

*Table 3. Permitted values for SSL/TLS protocols*

| **Value** | **Protocol** |
|---|---|
| SSLv3 | SSL 3.0 |
| TLSv1 | TLS 1.0 |
| TLSv1.1 | TLS 1.1 |
| TLSv1.2 | TLS 1.2 |

Use the entry listed in the **Value** column in the route property. The corresponding entry in the **Protocol** column is for information only.

**SSLClientSiteDN_C**
Use this property to specify a country name to select a certificate to send to the SSL/TLS server. Certificate matching is not case sensitive. If you do not specify this property, certificates are accepted with any country name. If you change this property (and **SSLClient** is set to true), the route is stopped, and restarted when a refresh command is issued. All connections to the route are stopped.

**SSLClientSiteDN_CN**
Use this property to specify a common name to select a certificate to send to the SSL/TLS server. Certificate matching is not case sensitive. If you do not specify this property, certificates are accepted with any common name. If you change this property (and **SSLClient** is set to true), the route is stopped, and restarted when a refresh command is issued. All connections to the route are stopped.

**SSLClientSiteDN_DC**
Use this property to specify a domain component name to select a certificate to send to the SSL/TLS server. Certificate matching is not case sensitive. You can specify multiple DCs by separating them with commas. Each DC represents an element in a domain name, for example the domain name `example.ibm.com` is represented as `example,ibm,com` using commas to separate the multiple values. If you do not specify this property, certificates are accepted with any domain component name. If you change this property (and **SSLClient** is set to `true`), the route is stopped, and restarted when a refresh command is issued. All connections to the route are stopped.

**SSLClientSiteDN_DNQ**
Use this property to specify a domain qualifier to select a certificate to send to the SSL/TLS server. Certificate matching is not case sensitive. If you do not specify this property, certificates are accepted with any domain qualifier. If you change this property (and **SSLClient** is set to `true`), the route is stopped, and restarted when a refresh command is issued. All connections to the route are stopped.

**SSLClientSiteDN_L**
Use this property to specify a Location name to select a certificate to send to the SSL/TLS server. Certificate matching is not case sensitive. If you do not specify this property, certificates are accepted with any location name. If you change this property (and **SSLClient** is set to `true`), the route is stopped, and restarted when a refresh command is issued. All connections to the route are stopped.

**SSLClientSiteDN_O**
Use this property to specify an Organization name to select a certificate to send to the SSL/TLS server. Certificate matching is not case sensitive. If you do not specify this property, certificates are accepted with any organization name. If you change this property (and **SSLClient** is set to `true`), the route is stopped, and restarted when a refresh command is issued. All connections to the route are stopped.

**SSLClientSiteDN_OU**
Use this property to specify an Organizational Unit (OU) name to select a certificate to send to the SSL/TLS server. You can specify multiple OUs by separating them with commas. (Match a literal comma by prefixing it with a backslash (\) character.) Certificate matching is not case sensitive. If you do not specify this property, certificates are accepted with any OU name. If you change this property (and **SSLClient** is set to `true`), the route is stopped and restarted when a refresh command is issued. All connections to this route are stopped.

**SSLClientSiteDN_PC**
Use this property to specify a postal code to select a certificate to send to the SSL/TLS server. Certificate matching is not case sensitive. If you do not specify this property, certificates are accepted with any postal code. If you change this property (and **SSLClient** is set to `true`), the route is stopped, and restarted when a refresh command is issued. All connections to the route are stopped.

**SSLClientSiteDN_ST**
Use this property to specify a State name to select a certificate to send to the SSL/TLS server. Certificate matching is not case sensitive. If you do not specify this property, certificates are accepted with any state name. If you change this property (and **SSLClient** is set to `true`), the route is stopped, and restarted when a refresh command is issued. All connections to the route are stopped.

**SSLClientSiteDN_Street**
Use this property to specify a street name to select a certificate to send to the SSL/TLS server. Certificate matching is not case sensitive. If you do not specify this property, certificates are accepted with any street name. If you change this property (and **SSLClient** is set to `true`), the route is stopped, and restarted when a refresh command is issued. All connections to the route are stopped.

**SSLClientSiteDN_T**
Use this property to specify a title to select a certificate to send to the SSL/TLS server. Certificate matching is not case sensitive. If you do not specify this property, certificates are accepted with any title. If you change this property (and **SSLClient** is set to `true`), the route is stopped, and restarted when a refresh command is issued. All connections to the route are stopped.

**SSLClientSiteDN_UID**
Use this property to specify a user ID to select a certificate to send to the SSL/TLS server. Certificate matching is not case sensitive. If you do not specify this property, certificates are accepted with any user ID. If you change this property (and **SSLClient** is set to `true`), the route is stopped, and restarted when a refresh command is issued. All connections to the route are stopped.

**SSLClientSiteLabel**
Use this property to specify a label name to select a certificate to send to the SSL/TLS server. If you do not specify this property, certificates are accepted with any label name. If you change this property (and **SSLClient** is set to `true`), the route is stopped, and restarted when a refresh command is issued. All connections to the route are stopped.

**SSLExitData**
Use this property to provide a user-defined string to be passed to the exit.

**SSLExitName**
Use this property to define the class name for the exit that will be called when the route is acting as an SSL/TLS client or an SSL/TLS server. The name must include any package name; for example, `com.ibm.mq.ipt.exit.TestExit`.

**SSLExitPath**
Use this property to define the location of the exit to be used to load a copy of the exit. The name must be a fully qualified name to be used to locate the class file or the name of a `.jar` file that contains the class file; for example, `C:\mqipt\exits` or `C:\mqipt\exits\exits.jar`.

**SSLExitTimeout**
Use this property to define how long MQIPT waits for the exit to complete before terminating the connection request. A value of `0` means that MQIPT waits indefinitely.

**SSLPlainConnections**
Use this property to specify whether SSL/TLS is mandatory for connections to the MQIPT listener port of a route configured to accept inbound SSL/TLS connections. This property is applicable to routes that have either the **SSLServer** or **SSLProxyMode** property set to true. If enabled, this property allows unencrypted connections to connect to the route listener port, which means that MQIPT can forward all IBM MQ connections to the queue manager's listener port regardless of whether the connection is encrypted. If you do not set this parameter, or set it to `false`, only inbound SSL/TLS connections are allowed. If you change this property, the route is stopped, and restarted when a refresh command is issued. All connections to the route are stopped.

**SSLProxyMode**
: Set this property to `true` to make the route accept only SSL/TLS client connection requests and to tunnel the request directly to the destination. If you change this property, the route is stopped and restarted when a refresh command is issued. All connections to this route are stopped. This property cannot be used in conjunction with the following properties:
  - **SocksClient**
  - **SSLClient**
  - **SSLServer**

**SSLServer**
: Set this property to `true` to make the route act as an SSL/TLS server and accept incoming SSL/TLS connections. Setting **SSLServer** to `true` implies that the caller is another MQIPT acting as an SSL/TLS client, or is a IBM MQ client or queue manager with SSL/TLS enabled. If you change this property, the route is stopped, and restarted when a refresh command is issued. All connections to the route are stopped. This property cannot be used in conjunction with the following properties:
  - **SocksServer**
  - **SSLProxyMode**

**SSLServerCAKeyRing**
: The fully-qualified file name of the key-ring file containing CA certificates, used to authenticate certificates from the SSL/TLS client. On Windows platforms, you must use a double backslash (\\) as the file separator. If you change this property (and **SSLServer** is set to `true`), the route is stopped, and restarted when a refresh command is issued. All connections to this route are stopped.

**SSLServerCAKeyRingPW**
: The fully-qualified file name containing the password to open the server CA key-ring file. On Windows platforms, you must use a double backslash (\\) as the file separator. If you change this property (and **SSLServer** is set to `true`), the route is stopped, and restarted when a refresh command is issued. All connections to this route are stopped.

**SSLServerAskClientAuth**
: Use this property to request SSL/TLS client authentication by the SSL/TLS server. The SSL/TLS client must have its own certificate to send to the SSL/TLS server. The certificate is retrieved from the key-ring file. If you change this property (and **SSLServer** is set to `true`), the route is stopped, and restarted when a refresh command is issued. All connections to this route are stopped.

**SSLServerCipherSuites**
: The name of the SSL/TLS CipherSuite to use on the SSL/TLS server side. This can be one or more of the supported CipherSuites. If you leave this blank, the SSL/TLS server uses the supported CipherSuites from the **SSLServerKeyRing** property. If you change this property (and **SSLServer** is set to `true`), the route is stopped, and restarted when a refresh command is issued. All connections to this route are stopped.

**SSLServerDN_C**
: Use this property to accept certificates received from the SSL/TLS client of this country name. The name can be prefixed or suffixed with an asterisk (*) to extend its scope. Certificate matching is not case sensitive. If you do not specify this property, certificates are accepted with any company name. If you

change this property (and **SSLServer** is set to `true`), the route is stopped, and restarted when a refresh command is issued. All connections to the route are stopped.

**SSLServerDN_CN**
Use this property to accept certificates received from the SSL/TLS client of this common name. The name can be prefixed or suffixed with an asterisk (*) to extend its scope. Certificate matching is not case sensitive. If you do not specify this property, certificates are accepted with any common name. If you change this property (and **SSLServer** is set to `true`), the route is stopped, and restarted when a refresh command is issued. All connections to the route are stopped.

**SSLServerDN_DC**
Use this property to accept certificates received from the SSL/TLS client of this domain component name. The name can be prefixed or suffixed with an asterisk (*) to extend its scope. Certificate matching is not case sensitive. You can specify multiple DCs by separating them with commas. Each DC represents an element in a domain name, for example the domain name `example.ibm.com` is represented as `example,ibm,com` using commas to separate the multiple values. If you do not specify this property, certificates are accepted with any domain component name. If you change this property (and **SSLServer** is set to `true`), the route is stopped, and restarted when a refresh command is issued. All connections to the route are stopped.

**SSLServerDN_DNQ**
Use this property to accept certificates received from the SSL/TLS client of this domain qualifier. The name can be prefixed or suffixed with an asterisk (*) to extend its scope. Certificate matching is not case sensitive. If you do not specify this property, certificates are accepted with any domain qualifier. If you change this property (and **SSLServer** is set to `true`), the route is stopped, and restarted when a refresh command is issued. All connections to the route are stopped.

**SSLServerDN_L**
Use this property to accept certificates received from the SSL/TLS client of this location. The name can be prefixed or suffixed with an asterisk (*) to extend its scope. Certificate matching is not case sensitive. If you do not specify this property, certificates are accepted with any location. If you change this property (and **SSLServer** is set to `true`), the route is stopped, and restarted when a refresh command is issued. All connections to the route are stopped.

**SSLServerDN_O**
Use this property to accept certificates received from the SSL/TLS client of this organization. The name can be prefixed or suffixed with an asterisk (*) to extend its scope. Certificate matching is not case sensitive. If you do not specify this property, certificates are accepted with any organization. If you change this property (and **SSLServer** is set to `true`), the route is stopped, and restarted when a refresh command is issued. All connections to the route are stopped.

**SSLServerDN_OU**
Use this property to accept certificates received from the SSL/TLS client of this Organizational Unit (OU). The name can be prefixed or suffixed with an asterisk (*) to extend its scope. You can specify multiple OUs by separating them with commas. (Match a literal comma by prefixing it with a backslash (\) character.) Certificate matching is not case sensitive. If you do not specify this property, certificates are accepted with any OU name. If you change this

property (and **SSLServer** is set to `true`), the route is stopped and restarted when a refresh command is issued. All connections to this route are stopped.

**SSLServerDN_PC**
Use this property to accept certificates received from the SSL/TLS client of this postal code. The name can be prefixed or suffixed with an asterisk (*) to extend its scope. Certificate matching is not case sensitive. If you do not specify this property, certificates are accepted with any postal code. If you change this property (and **SSLServer** is set to `true`), the route is stopped, and restarted when a refresh command is issued. All connections to the route are stopped.

**SSLServerDN_ST**
Use this property to accept certificates received from the SSL/TLS client of this state. The name can be prefixed or suffixed with an asterisk (*) to extend its scope. Certificate matching is not case sensitive. If you do not specify this property, certificates are accepted with any state. If you change this property (and **SSLServer** is set to `true`), the route is stopped, and restarted when a refresh command is issued. All connections to the route are stopped.

**SSLServerDN_Street**
Use this property to accept certificates received from the SSL/TLS client of this street name. The name can be prefixed or suffixed with an asterisk (*) to extend its scope. Certificate matching is not case sensitive. If you do not specify this property, certificates are accepted with any street name. If you change this property (and **SSLServer** is set to `true`), the route is stopped, and restarted when a refresh command is issued. All connections to the route are stopped.

**SSLServerDN_T**
Use this property to accept certificates received from the SSL/TLS client of this title. The name can be prefixed or suffixed with an asterisk (*) to extend its scope. Certificate matching is not case sensitive. If you do not specify this property, certificates are accepted with any title. If you change this property (and **SSLServer** is set to `true`), the route is stopped, and restarted when a refresh command is issued. All connections to the route are stopped.

**SSLServerDN_UID**
Use this property to accept certificates received from the SSL/TLS client of this user ID. The name can be prefixed or suffixed with an asterisk (*) to extend its scope. Certificate matching is not case sensitive. If you do not specify this property, certificates are accepted with any user ID. If you change this property (and **SSLServer** is set to `true`), the route is stopped, and restarted when a refresh command is issued. All connections to the route are stopped.

**SSLServerExit**
Use this property to enable or disable the use of an exit when the route is acting as an SSL/TLS server. This allows you to define exit details in the configuration file without them actually being used.

**SSLServerKeyRing**
The fully-qualified file name of the key-ring file containing the server certificate; on Windows platforms, you must use a double backslash (\\) as the file separator. You must specify **SSLServerKeyRing** if you set **SSLServer** to `true`. If you change this **SSLServerKeyRing** property (and **SSLServer** is set to `true`), the route is stopped, and restarted when a refresh command is issued. All connections to the route are stopped.

**SSLServerKeyRingPW**
The fully-qualified file name containing the password to open the server key-ring file; on Windows platforms, you must use a double backslash (\\) as

the file separator. You must specify **SSLServerKeyRingPW** if you set **SSLServer** to `true`. If you change this **SSLServerKeyRingPW** property (and **SSLServer** is set to `true`), the route is stopped, and restarted when a refresh command is issued. All connections to the route are stopped.

**SSLServerProtocols**
Used to restrict the set of enabled secure socket protocols that are used to accept inbound connections to the route listener port for a route when **SSLClient** is set to `true`).

You can specify multiple values by separating them with commas. If you do not specify this property, all supported JSSE protocols are enabled by default with the exception of SSL 3.0. From version 2.1.0.2 of MQIPT, you must explicitly set `SSLv3` on this property for SSL 3.0 to be enabled.

*Table 4. Permitted values for SSL/TLS protocols*

| **Value** | **Protocol** |
|---|---|
| `SSLv3` | SSL 3.0 |
| `TLSv1` | TLS 1.0 |
| `TLSv1.1` | TLS 1.1 |
| `TLSv1.2` | TLS 1.2 |

Use the entry listed in the **Value** column in the route property. The corresponding entry in the **Protocol** column is for information only.

**SSLServerSiteDN_C**
Use this property to specify a country name to select a certificate to send to the SSL/TLS client. Certificate matching is not case sensitive. If you do not specify this property, certificates are accepted with any country name. If you change this property (and **SSLServer** is set to `true`), the route is stopped, and restarted when a refresh command is issued. All connections to the route are stopped.

**SSLServerSiteDN_CN**
Use this property to specify a Common Name to select a certificate to send to the SSL/TLS client. Certificate matching is not case sensitive. If you do not specify this property, certificates are accepted with any common name. If you change this property (and **SSLServer** is set to `true`), the route is stopped, and restarted when a refresh command is issued. All connections to the route are stopped.

**SSLServerSiteDN_DC**
Use this property to specify a domain component name to select a certificate to send to the SSL/TLS client. Certificate matching is not case sensitive. You can specify multiple DCs by separating them with commas. Each DC represents an element in a domain name, for example the domain name `example.ibm.com` is represented as `example,ibm,com` using commas to separate the multiple values. If you do not specify this property, certificates are accepted with any domain component name. If you change this property (and **SSLServer** is set to `true`), the route is stopped, and restarted when a refresh command is issued. All connections to the route are stopped.

**SSLServerSiteDN_DNQ**
Use this property to specify a domain qualifier to select a certificate to send to the SSL/TLS client. Certificate matching is not case sensitive. If you do not specify this property, certificates are accepted with any domain qualifier. If you change this property (and **SSLServer** is set to `true`), the route is stopped, and restarted when a refresh command is issued. All connections to the route are stopped.

**SSLServerSiteDN_L**
: Use this property to specify a Location name to select a certificate to send to the SSL/TLS client. Certificate matching is not case sensitive. If you do not specify this property, certificates are accepted with any location name. If you change this property (and **SSLServer** is set to `true`), the route is stopped, and restarted when a refresh command is issued. All connections to the route are stopped.

**SSLServerSiteDN_O**
: Use this property to specify an organization name to select a certificate to send to the SSL/TLS client. Certificate matching is not case sensitive. If you do not specify this property, certificates are accepted with any organization name. If you change this property (and **SSLServer** is set to `true`), the route is stopped, and restarted when a refresh command is issued. All connections to the route are stopped.

**SSLServerSiteDN_OU**
: Use this property to specify an Organizational Unit (OU) name to select a certificate to send to the SSL/TLS client. You can specify multiple OUs by separating them with commas. (Match a literal comma by prefixing it with a backslash (\) character.) Certificate matching is not case sensitive. If you do not specify this property, certificates are accepted with any OU name. If you change this property (and **SSLServer** is set to `true`), the route is stopped and restarted when a refresh command is issued. All connections to this route are stopped.

**SSLServerSiteDN_PC**
: Use this property to specify a postal code to select a certificate to send to the SSL/TLS client. Certificate matching is not case sensitive. If you do not specify this property, certificates are accepted with any postal code. If you change this property (and **SSLServer** is set to `true`), the route is stopped, and restarted when a refresh command is issued. All connections to the route are stopped.

**SSLServerSiteDN_ST**
: Use this property to specify a State name to select a certificate to send to the SSL/TLS client. Certificate matching is not case sensitive. If you do not specify this property, certificates are accepted with any state name. If you change this property (and **SSLServer** is set to `true`), the route is stopped, and restarted when a refresh command is issued. All connections to the route are stopped.

**SSLServerSiteDN_Street**
: Use this property to specify a street name to select a certificate to send to the SSL/TLS client. Certificate matching is not case sensitive. If you do not specify this property, certificates are accepted with any street name. If you change this property (and **SSLServer** is set to `true`), the route is stopped, and restarted when a refresh command is issued. All connections to the route are stopped.

**SSLServerSiteDN_T**
: Use this property to specify a title to select a certificate to send to the SSL/TLS client. Certificate matching is not case sensitive. If you do not specify this property, certificates are accepted with any title. If you change this property (and **SSLServer** is set to `true`), the route is stopped, and restarted when a refresh command is issued. All connections to the route are stopped.

**SSLServerSiteDN_UID**
: Use this property to specify a user ID to select a certificate to send to the SSL/TLS client. Certificate matching is not case sensitive. If you do not specify this property, certificates are accepted with any user ID. If you change this

property (and **SSLServer** is set to `true`), the route is stopped, and restarted when a refresh command is issued. All connections to the route are stopped.

**SSLServerSiteLabel**
: Use this property to specify a label name to select a certificate to send to the SSL/TLS client. If you do not specify this property, certificates are accepted with any label name. If you change this property (and **SSLServer** is set to `true`), the route is stopped, and restarted when a refresh command is issued. All connections to the route are stopped.

**TCPKeepAlive**
: Set this property to `true` to enable the sending of TCP/IP keep-alive packets periodically to prevent the connections on this route becoming idle. This reduces the chances of the MQIPT connections being severed by a firewall or router. The sending of TCP/IP keep-alive packets is controlled by operating system tuning parameters; consult your operating system documentation for further details on how to tune keep-alive. If you do not set this parameter, or set it to `false`, keep-alive packets are not sent.

**Trace**
: The level of tracing required for this route. Enabling trace for one route does not enable trace for any other routes. If you need to trace more than one route, you must add the **Trace** property to the `[route]` section of each route to be traced. This property should be an integer in the range `0 - 5`, where `0` indicates that trace is disabled, and any other value indicates that trace is enabled. The default value is `0`. If the `[route]` section does not include a **Trace** property, the **Trace** property from the `[global]` section is used. For information about tracing threads that are not associated with a route, see **Trace** in the `[global]` section. If a change to this property affects a route, the new value is used when the refresh command is issued. All connections use the new value immediately. The route is not stopped.

**UriName**
: This property can be used to change the name of the Uniform Resource Identifier of the resource when using an HTTP proxy, although the default value will suffice for most configurations:

```
HTTP://destination:destination_port/mqipt
```

If you change this property (and **HTTP** is set to `true`), the route is stopped, and restarted when a refresh command is issued.

## Making backups

There are a number of MQIPT files that you should back up as part of your regular backup procedures.

Back up the following files on a regular basis:
- The configuration file, `mqipt.conf`
- The SSL/TLS key-ring files in `mqipt.conf` as defined by the following properties:
  - **SSLClientKeyRing**
  - **SSLClientCAKeyRing**
  - **SSLServerKeyRing**
  - **SSLServerCAKeyRing**
- The SSL/TLS key-ring password files in `mqipt.conf` as defined by the following properties:

- **SSLClientKeyRingPW**
- **SSLClientCAKeyRingPW**
- **SSLServerKeyRingPW**
- **SSLServerCAKeyRingPW**

- The Administration Client configuration file, `client.conf`, which contains connection information about all the instances of MQIPT known to the Administration Client.
- The policy file specified by the **SecurityManagerPolicy**, if that parameter has been set.
- The security exit files and certificate exit files as defined by the following properties:
  - **SecurityExitName**
  - **SSLExitName**

## Performance tuning

You can tune the relative performance of each route by using a combination of a thread pool and an idle timeout specification.

### Connection threads

ach MQIPT route is assigned a working pool of concurrently running threads that handle incoming communication requests. At initialization, a pool of threads is created (of the size specified in the route's `MinConnectionThreads` attribute), and a thread is assigned to handle the first incoming request. When this request arrives, another thread is assigned, ready for the next incoming request. When all threads are assigned for work, a new thread is created, added to the working pool, and assigned for work.

In this way, the pool grows until the maximum number of threads (specified in **MaxConnectionThreads**) is reached. Threads are released back to the pool when a conversation ends, or the specified idle timeout period has elapsed. When the maximum number of working threads is reached, the next incoming request waits until a thread is released back to the working pool.

You can reduce the time that requests might have to wait by increasing the number of available threads. However, you must balance this increase with the system resources that are available.

### Idle timeout

By default, working threads are not terminated because of inactivity. When a thread has been assigned to a conversation, it remains assigned to that conversation until it is closed normally, the route is deactivated, or MQIPT is shut down. Optionally, you can specify an idle timeout interval (in minutes) in the **IdleTimeout** property so that threads that have been inactive for the specified period of time are recycled. Threads are recycled for use by placing them back into the working pool.

If IBM MQ activity is intermittent, set its heartbeat interval to a value less than that of the MQIPT timeout so that threads are not constantly recycled.

# Chapter 8. Troubleshooting and support

There are a number of steps you can follow to help determine the nature of any problems you might encounter.

1. Check for the following common errors:
   - The **`HTTP`** property is set to `true` on a route directly connected to a queue manager.
   - The **`SSLClient`** property is set to `true` on a route directly connected to a queue manager that is not configured to use SSL/TLS.
   - The passwords stored for the key-ring files are case-sensitive.
2. If you find any FFST reports in the errors subdirectory, MQIPT was correctly installed but there might have been a problem with the configuration.

   Each FFST reports a problem that causes MQIPT or a route to terminate its startup process. Fix the problem that caused each FFST. Then delete the old FFST and restart or refresh MQIPT.
3. If there is not an FFST and there is no trace output, MQIPT has not been installed correctly. Check that all the files have been put in the correct place. To check this, try to start MQIPT manually:
   a. Open a command prompt. Go to the `bin` subdirectory and type:
      ```
      mqipt xxx
      ```
      where *`xxx`* is the MQIPT home directory; in this case, it is `..`.
   b. When MQIPT starts, look for the configuration in the home directory. Look for any error messages and FFST instances in the `errors` subdirectory.
   c. Look at the text output from MQIPT for any error messages. Check for instances of FFST. Correct any errors.

      **Note:** MQIPT will not start if there is a problem in the `[global]` section of the configuration file. A route will not start if there is a problem in the `[route]` section of the configuration file.
4. If there is not an FFST but you do have trace output, configure the MQIPT connections (`ConnectionLog=true`) and make the sender attempt a connection. Then check that a connection from the host has been logged.
   - If a connection from the host has been logged, the sender has not been configured correctly.
   - If a connection has not been logged, check that MQIPT is configured to forward the message to the correct host and port. Then treat as a normal channel problem.

## Automatically starting MQIPT

If you install MQIPT as a Windows Service, or as a UNIX or Linux **`init.d`** system service, it starts when the system is started.

### On Windows systems

Always try starting MQIPT manually before installing it as a Windows Service, to confirm correct installation. See "Using a Windows service control program" on page 147 for more details.

If the MQIPT service does not start correctly, complete the following steps:

1. Open the Windows Registry Editor and navigate to the `HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\services\MQInternetPassThru` key. Check that the **ConfigFilePath** setting contains the correct path to the `mqipt.conf` configuration file. Also, check that the **ImagePath** setting contains the correct path to `mqiptService.exe`.
2. Run the **mqiptService** `-debugevents` command from an Administrator Command Prompt to write service startup information in the Windows application event log. Additional information is also displayed in the Command Prompt console window. Examine the diagnostic information to determine the cause of the failure.
3. If the cause of the failure is still not clear, use Windows Explorer to navigate to the directory specified in **ConfigFilePath** where `mqipt.conf` is located. Examine the contents of the errors subdirectory to look for FDC files containing FFST records.
4. If the cause of the failure is still not clear, enable trace by setting the **Trace** property to `5` in the `[global]` section of `mqipt.conf`. Restart the MQIPT service. A trace file is be written in the MQIPT errors directory. If necessary, contact IBM Software Support and supply the trace file along with any FDC files and the diagnostic output from the **mqiptService** `-debugevents` command.

## On UNIX and Linux systems

Always try starting MQIPT manually before installing it as a service, to confirm correct installation. See "Using a UNIX or Linux init.d system service" on page 148 for more details.

If the MQIPT service does not start correctly, complete the following steps as the root user:

1. Check that the MQIPT service is installed. You might need to uninstall and reinstall the service. To check that the service is installed:
   a. On AIX, run `lsitab mqipt` and check that the output shows the correct installation directory. Here is an example of the output for an MQIPT service running from the `/usr/opt/mqipt` installation:
   ```
   mqipt:2:once:/usr/opt/mqipt/bin/mqipt /usr/opt/mqipt > /dev/console 2>&1
   ```
   Check that the MQIPT executable named exists and is executable by the root user.
   b. On HP-UX, Linux and Solaris, check for the existence of the MQIPT **init.d** script. This is named `/sbin/init.d/mqipt` on HP-UX and `/etc/init.d/mqipt` on Linux and Solaris. The script must exist and must be executable by the root user.
2. Ensure that the installation directory contains the `mqipt.conf` file, which must be readable by the root user.
3. Check the output from the MQIPT startup.
   - On AIX, the MQIPT output is sent to `/dev/console`.
   - On HP-UX, Linux and Solaris platforms, the output is sent to a file named `console.log` in the `logs` directory of the MQIPT installation.

   Look for any MQIPT errors and address the cause. If no console output is present then MQIPT was not started by the operating system. Consult your operating system documentation for details of how to diagnose service startup failures.

4. If the cause of the failure is still not clear, navigate to the MQIPT installation directory where `mqipt.conf` is located. Examine the contents of the errors subdirectory to look for FDC files containing FFST records.
5. If the cause of the failure is still not clear, enable trace by setting the Trace property to 5 in the `[global]` section of `mqipt.conf`. Restart the MQIPT service. A trace file is written in the MQIPT errors directory. If necessary, contact IBM Software Support and supply the trace file along with any FDC files and the diagnostic output from `/dev/console` (AIX) or console.log (HP-UX, Linux and Solaris).

## Using a Windows service control program

A separate service control program, **`mqiptService.exe`**, is provided so that you can manage and start MQIPT as a Windows service.

If you want to run MQIPT as a system service, you can install only one such service on each system. You cannot install more than one MQIPT service on the same system, either from the same MQIPT installation or from different installations.

You must run the **`mqiptService`** command from an administrator command prompt in order to ensure you have the authority required to configure Windows services. Typically:

1. **start** > **Programs** > **Accessories** > **Command Prompt**
2. Right-click **Command Prompt** and select **Run as administrator**. At the command prompt, change directory to the MQIPT installation directory, for example:
   ```
   cd /D C:\mqipt\bin
   ```
3. Run **`mqiptService.exe`** from this command prompt.

**`mqiptService.exe`** takes the following command line arguments:

**`mqiptService -install`** ***`mqipt_location`***

Installs and registers the service, so that it appears on the Windows services panel as an automatic service.

You must reboot Windows after installing this service.

The path parameter, which must be supplied, is the fully-qualified path to the directory containing the `mqipt.conf` configuration file. Enclose the path in double quotation marks (") if it contains spaces.

**`mqiptService -remove`**

Removes the service so that it no longer appears on the Windows services panel.

**Note:** Only the installation of MQIPT that installed the service can be used to remove it. For example, if you have two MQIPT installations, one in `C:\MQIPT1` and one in `C:\mqipt2`, and you run the command `C:\MQIPT1\bin\mqiptService -install C:\mqipt1`, then only the **`mqiptService`** command from the `C:\MQIPT1` installation can subsequently be used to remove the service. Attempting to remove the service using a different installation causes error MQCPE083.

**`mqiptService ?`**

Displays help messages listing the valid arguments.

Specifying both **`-install`** and **`-remove`** on the same command causes an error.

If you call the `mqiptService.exe` program from the command line with no arguments, the program times out and returns with an error.

When the MQIPT service is started, all active MQIPT routes start. When the service is stopped, all routes are subjected to immediate shutdown.

### Using a UNIX or Linux init.d system service

A separate service control program, **`mqiptService`**, is provided so that you can manage and start MQIPT as a UNIX or Linux init.d system service that starts when the system boots.

If you want to run MQIPT as a system service, you can install only one such service on each system. You cannot install more than one MQIPT service on the same system, either from the same MQIPT installation or from different installations.

You must run the **`mqiptService`** command as root in order to ensure you have the authority required to configure services.

**`mqiptService`** takes the following command line arguments:

**`mqiptService -install`**

Installs and registers the service.

The `mqipt.conf` file for the service must be located in the top-level MQIPT installation directory of the installation from which you ran **`mqiptService`**.

Installing the service does not automatically start it. The service starts the next time the system is restarted. Consult your operating system service documentation if you need to start the MQIPT service immediately, without restarting.

**`mqiptService -remove`**

Removes the service so that it no longer starts at system boot time.

**Note:** Only the installation of MQIPT that installed the service can be used to remove it. For example, if you have two MQIPT installations, one in `/opt/mqipt` and one in `/usr/local/mqipt`, and you run the command `/opt/mqipt/bin/mqiptService -install`, then only the **`mqiptService`** command from the `/opt/mqipt` installation can subsequently be used to remove the service. Attempting to remove the service using a different installation causes error MQCPE083.

**`mqiptService ?`**

Displays help messages listing the valid arguments.

Specifying both **`-install`** and **`-remove`** on the same command causes an error.

When the MQIPT service is started, all active MQIPT routes start. When the service is stopped, all routes are subjected to immediate shutdown.

## Checking for end-to-end connectivity

If you cannot make a connection, check the connection log to see if the routes are set up correctly.

Create the connection log: In the mqipt.conf configuration file, set the **ConnectionLog** property to true. Start or refresh MQIPT, and attempt a connection. See “Connection logs” on page 58 for details.

1. If the connection log is not created in the logs directory below the home directory, MQIPT has not been installed correctly.
2. If no connection attempts are recorded, the sender has not been set up correctly.
3. If attempts are recorded, check that MQIPT is forwarding the messages to the correct address.

## Using JRE diagnostic options

In some cases you might need to use diagnostic functions that are built into the Java Runtime Environment (JRE). You should usually only do this under the direction of your IBM Software Support representative, as some diagnostic settings might impair normal MQIPT operation.

The **MQIPT_JVM_OPTIONS** environment variable can be used to pass diagnostic options to the underlying MQIPT JRE via the command line. All command parameters that are valid for the IBM JRE supplied with MQIPT can be used.

There are two common diagnostic options that can be used are:

**-Djavax.net.debug=all**
: This option enables diagnostics for SSL/TLS and network throughput. Setting this option causes a detailed log of internal network operations to be written to the console where MQIPT was started. This is particularly useful for debugging SSL/TLS handshake errors on routes with **SSLClient** or **SSLServer** set to true.

**-Djava.security.debug=access,failure**
: This option enables diagnostics for the Java Security Manager policy, for MQIPT instances with **SecurityManager** set to true. Setting this option causes a detailed log of security activities and their required permissions to be written to the console where MQIPT was started. It can be used to identify missing permissions in the policy file.

Here is an example of enabling both of these settings on Windows platforms:

```
set MQIPT_JVM_OPTIONS=-Djavax.net.debug=all -Djava.security.debug=access,failure
```

Here is an example of enabling both of these settings on UNIX and Linux platforms:

```
MQIPT_JVM_OPTIONS="-Djavax.net.debug=all -Djava.security.debug=access,failure"
export MQIPT_JVM_OPTIONS
```

For these settings to take effect, you must restart MQIPT from the command prompt where the environment variable is set.

For another use of **MQIPT_JVM_OPTIONS** when diagnosing problems, see “Tracing errors in iKeyman and iKeycmd” on page 150.

## Tracing errors in MQIPT

MQIPT provides a detailed execution trace facility, which is controlled by the **Trace** property.

Trace files are written to the *mqipt_home*\errors directory (where *mqipt_home* is the MQIPT home directory, which contains mqipt.conf). Each trace file produced has a name with the following format:

```
AMQyyyymmddnnnnnnnnn.n.TRC
```

Unexpected fatal errors are written as FFST records in an error log file located in the *mqipt_home*\errors directory. The FFST files have the following format:

```
AMQyyyymmddnnnnnnnnn.n.FDC
```

To enable trace, add the **Trace** configuration property to the appropriate section in the mqipt.conf file. From MQIPT V2.1, the **Trace** property is route-specific. This means that if you want to trace more than one route, you must add the **Trace** property to the [route] section of each route that you want to trace. Alternatively, if you add the **Trace** property to the [global] section, it is inherited by all routes that do not specify a **Trace** property.

## Tracing errors in iKeyman and iKeycmd

The **mqiptKeycmd** and **mqiptKeyman** commands have an execution trace facility which can diagnose errors in the certificate management tools.

To enable trace for these commands, set the following environment variable before running the mqiptKeycmd or mqiptKeyman command:

- On Windows systems:
  ```
  set MQIPT_JVM_OPTIONS=-Dkeyman.debug=true -Dkeyman.logging=true
  ```
- On UNIX and Linux systems:
  ```
  MQIPT_JVM_OPTIONS="-Dkeyman.debug=true -Dkeyman.logging=true"
  export MQIPT_JVM_OPTIONS
  ```

A trace file is created in the current working directory. The trace file name has the following format:

```
debugTrace.n
```

where *n* is an incrementing number starting at 0.

The user running the certificate management tool must have permission to create files in the current working directory, otherwise the command fails with an error.

After you have finished recording trace logs, unset the environment variable.

## Reporting problems

If you need to report a problem to the IBM® Service Center, send relevant information that will help to resolve the problem more quickly.

Complete the following steps to obtain the required information:

1. Synchronize the system clock on each computer involved, including all those running IBM MQ and MQIPT. This operation helps to match trace entries in different trace files.
2. Move old trace files to a backup directory so that new trace files contain information related only to this problem.
3. Run the client to reproduce the problem and create new trace files.
4. Send a copy of all MQIPT .TRC, .FDC, and .log files. Also send a simple network diagram of all the computers used between the IBM MQ endpoints,

including firewalls, routers, load balancers, and servers. For each computer, include its name, IP address, and relevant port numbers.

# Chapter 9. Messages

When run from the command line, MQIPT displays information, warning, and error messages on the console.

All message identifiers follow the same format:

`MQC`*`psnnn`*

where:

- *p* is the producer of the message:
  - `A`: Administration Client
  - `P`: MQIPT
- *s* is the severity of the message:
  - `I`: information
  - `W`: warning
  - `E`: error
- *nnn* is the three-digit message number.

## List of MQIPT MQC messages

**MQCAE001 Unknown host: {0}**

**Explanation:** The MQIPT host cannot be found.

**User response:** Check the host name was specified correctly. Try to PING the host name or use its IP address.

**MQCAE002 The following error was reported by the system: {0}**

**Explanation:** An error has occurred while communicating with an MQIPT.

**User response:** Review the text of the error message and take the appropriate action.

**MQCAE005 No valid destination address has been defined**

**Explanation:** When adding a route, the destination field has been left blank.

**User response:** Enter a valid destination address.

**MQCAE006 No valid destination port has been defined**

**Explanation:** When adding a route, the destination port address field has been left blank.

**User response:** Enter a valid destination port address.

**MQCAE007 No valid listener port has been defined**

**Explanation:** When adding a route, the listener port address field has been left blank.

**User response:** Enter a valid listener port address, between 1 - 65535.

**MQCAE008 No valid network address has been defined**

**Explanation:** When adding an MQIPT, the network address field has been left blank.

**User response:** Enter a valid network address.

**MQCAE009 No valid command port has been defined**

**Explanation:** When adding an MQIPT, an invalid command port address has been used.

**User response:** Enter a valid command port address, between 1 - 65535.

**MQCAE010 Could not show online help**

**Explanation:** The file for online help was available but could not be displayed.

**User response:** Make sure that Netscape is installed and available in the system PATH environment variable.

**MQCAE011 Could not parse parameter**

**Explanation:** There has been an internal error that caused an attempt to be made to update a nonexistent parameter in the table.

**User response:** If the condition persists contact IBM Software Support.

**MQCAE012 Could not find online help file {0}**

**Explanation:** File "passtfrm.htm" could not be found.

**User response:** Make sure this file is accessible in the doc language subdirectory.

**MQCAE013 Interrupted while trying to show online help**

**Explanation:** A system error occurred while displaying the online help.

**User response:** Try again. Contact IBM Software Support if the condition persists.

**MQCAE015 The password you have just entered has not been recognized**

**Explanation:** The MQIPT expects a valid password, and the one used in the last command was incorrect. It must match that defined in the configuration file.

**User response:** Change the password using the menu MQIPT->Connection panel and retry the last command again.

**MQCAE016 Node mismatch**

**Explanation:** There is an internal inconsistency between the node selected on the tree and the data held in memory.

**User response:** Close the Administration Client and retry the command again. Contact IBM Software Support if the condition persists.

**MQCAE017 Could not create NLS text for message {0}**

**Explanation:** No NLS text has been found for the defined message number.

**User response:** The guiadmin.properties file may have become corrupted and the specified message number could not be found. Check the following:

1. look in the Readme file for a possible new message
2. guiadmin.properties file is in the guiadmin.jar file
3. the message number is in the guiadmin.properties file

**MQCAE019 You have failed to repeat your proposed new password**

**Explanation:** When changing the password, it has not been entered twice, for verification.

**User response:** Enter the new password again in the appropriate field.

**MQCAE020 Failed to change MQIPT access parameters**

**Explanation:** An internal error has been detected while trying to change MQIPT access parameters.

**User response:** Close the Administration Client and try the command again. If the condition persists contact IBM Software Support.

**MQCAE021 Internal failure to identify MQIPT**

**Explanation:** An internal error has been detected while trying to save a configuration file on an MQIPT.

**User response:** Close the Administration Client and try the command again. If the condition persists contact IBM Software Support.

**MQCAE022 Internal failure to save MQIPT configuration**

**Explanation:** An internal error has been detected while trying to save a configuration file on an MQIPT.

**User response:** Close the Administration Client and try the command again. If the condition persists contact IBM Software Support.

**MQCAE023 MQIPT {0} did not recognize your password**

**Explanation:** The MQIPT expects a valid password, and the one used in the last command was incorrect. It must match that defined in the configuration file.

**User response:** Change the password using the menu MQIPT->Connection panel and retry the last command again.

**MQCAE024 MQIPT {0} has not recognized the command**

**Explanation:** The Administration Client has sent a command to the MQIPT which it has not recognized.

**User response:** Make sure the version of code used by the Administration Client is the same as the MQIPT.

**MQCAE025 MQIPT {0} has failed to send configuration file**

**Explanation:** The MQIPT attempted to send the configuration file, but failed.

**User response:** Close the Administration Client and retry the command. If this still fails, stop and restart the MQIPT.

**MQCAE026 Remote shutdown is disabled on MQIPT {0}**

**Explanation:** An attempt to shut down the MQIPT remotely has failed because remote shutdown was not enabled in the configuration file.

**User response:** To enable remote shutdown of MQIPT, edit the configuration file and set the RemotShutDown property to true.

**MQCAE027 Look and feel {0} is not supported**

**Explanation:** The recommended look and feel for the platform you are using is not available.

**User response:** Processing continues with the system default look and feel.

**MQCAE028 Look and feel class {0} cannot be found**

**Explanation:** The recommended look and feel for the platform you are using is not available.

**User response:** Processing continues with the system default look and feel.

**MQCAE029 Must be non-negative and no bigger than Maximum Connection Threads**

**Explanation:** The Minimum Connection Threads value must be less than or equal to the Maximum Connection Threads value.

**User response:** Change the value accordingly.

**MQCAE030 Must be greater than zero and at least as big as Minimum Connection Threads**

**Explanation:** The Maximum Connection Threads value must be greater than the Minimum Connection Threads value.

**User response:** Change the value accordingly.

**MQCAE031 Port numbers must be in the range 0 to 65535**

**Explanation:** You are attempting to set a value that does not meet the specification.

**User response:** Change the value accordingly.

**MQCAE032 Trace must be in the range 0 to 5**

**Explanation:** Trace must be in the range 0 to 5

**User response:** Change the value accordingly.

**MQCAE033 Maximum Log Size must be in the range 5 to 50**

**Explanation:** Max Log file size must be in the range 5 to 50

**User response:** Change the value accordingly.

**MQCAE049 No route has been selected on any MQIPT**

**Explanation:** An attempt has been made to delete a route without first selecting the route to be deleted.

**User response:** Select a route and retry the command again.

**MQCAE050 Could not connect to MQIPT {0}**

**Explanation:** The Administration Client could not connect to the specified MQIPT.

**User response:** This can be caused by any of the following :

1. MQIPT is not running
2. MQIPT is not listening on its CommandPort
3. Only one Administration Client is using the MQIPT CommandPort
4. The request has timed out.

**MQCAE051 Could not read reply from MQIPT {0}**

**Explanation:** A reply was received from the MQIPT that did not conform to the expected protocol.

**User response:** Make sure the version of code used by the Administration Client is the same as the MQIPT.

**MQCAE052 Configuration has not been saved**

**Explanation:** A valid reply was received from the MQIPT but it subsequently failed to save the configuration file.

**User response:** This can be caused by any of the following :

1. MQIPT does not have write access to the configuration file
2. The configuration file has been opened by another process
3. The disk is full

**MQCAE053 MQIPT has not confirmed saving of configuration**

**Explanation:** The configuration file has been sent to the MQIPT but the MQIPT failed to acknowledge it.

**User response:** This can be caused by any of the following :

1. MQIPT is not running

2. MQIPT is not listening on its CommandPort
3. Only one Administration Client is using the MQIPT CommandPort
4. The request has timed out

**MQCAE054 MQIPT data has not been refreshed**

**Explanation:** Contact has been made with the MQIPT but the Administration Client was unable to read the configuration file.

**User response:** This can be caused by any of the following :

1. MQIPT has failed
2. The request has timed out

**MQCAE055 No MQIPT or route on an MQIPT has been selected**

**Explanation:** Your chosen menu option cannot be performed because no MQIPT or route has been selected.

**User response:** Select an MQIPT or route and try again.

**MQCAE056 Duplicate listener port has been rejected**

**Explanation:** The specified listener port has been rejected because it is already being used by another route.

**User response:** Choose another listener port address.

**MQCAI002 The MQIPT has been removed from display**

**Explanation:** The MQIPT whose node you selected on the tree has been removed from the client's memory.

**MQCAI003 New route added to the display**

**Explanation:** The new route that you have just specified has been added to the current MQIPT.

**MQCAI004 Route has been removed from the display**

**Explanation:** The route that you selected on the tree has been removed from the client's memory.

**MQCAI005 Selected MQIPT is being displayed**

**Explanation:** The global parameters of the MQIPT that you selected on the tree are being shown in the table.

**MQCAI006 Selected route is being displayed**

**Explanation:** The parameters of the route that you selected on the tree are being shown in the table.

**MQCAI007 Client configuration has been saved**

**Explanation:** The access parameters for all the MQIPTs on the tree have been saved.

**MQCAI008 Display of online help succeeded**

**Explanation:** The online help has been displayed as requested.

**MQCAI009 Table has been updated**

**Explanation:** The value you have just entered on the table has been used to update the model in memory.

**MQCAI010 No MQIPT or route has been selected**

**Explanation:** No action has been taken because there is insufficient information on which to act.

**MQCAI011 User action has been cancelled**

**Explanation:** You have cancelled out of an action, involving a popup window, that you had previously initiated.

**MQCAI014 Configuration has been saved on MQIPT**

**Explanation:** A new configuration file has been saved on the MQIPT that is currently selected on the tree, and it has been used to restart the MQIPT.

**MQCAI015 Online help has terminated**

**Explanation:** The online help has been displayed as requested and subsequently terminated.

**MQCAI017 Select File/Add MQIPT to add an MQIPT to the tree**

**Explanation:** This message appears when there are no MQIPTs on the tree; it tells you how to add one.

**MQCAI018 New MQIPT added to display**

**Explanation:** A new MQIPT has been added to the tree as instructed.

**MQCAI019 MQIPT access parameters have been changed**

**Explanation:** The access parameters of the MQIPT that is currently selected on the tree have been changed.

**MQCAI021 Select an MQIPT or route on the tree to display its contents**

**Explanation:** This message appears when no information is being shown on the table; it tells you how to see some.

**MQCAI022 The command port has changed**

**Explanation:** The MQIPT whose command port was instructed to change has now changed.

**MQCAI023 The password has changed**

**Explanation:** Any future communication with the MQIPT which you have just changed will use the new password.

**MQCAI025 MQIPT {0} has been refreshed**

**Explanation:** The information you hold on the MQIPT has been updated by reading its configuration file.

**MQCAI026 MQIPT {0} has received shutdown request**

**Explanation:** The MQIPT has acknowledged receipt of a shutdown request and will now shut down.

**MQCAI027 Client configuration has been refreshed**

**Explanation:** The information displayed in the Administration Client has been refreshed from the local file client.conf.

**MQCAI028 MQIPT {0} is active**

**Explanation:** The MQIPT has responded successfully to a ping request.

**MQCAI029 MQIPT {0} is not active**

**Explanation:** The MQIPT has not responded to a ping request within a specified time.

**User response:** This can be caused by any of the following:

1. MQIPT is not running
2. MQIPT is not listening on its CommandPort
3. The request has timed out. The timeout can be increased by changing the timeout property on the connection information for the MQIPT.

**MQCAI030 Route {0} is active**

**Explanation:** The MQIPT route has responded successfully to a ping request.

**MQCAI031 Route {0} is not active**

**Explanation:** The MQIPT route has not responded to a ping request within a specified time.

**User response:** This can be caused by any of the following :

1. MQIPT is not running
2. MQIPT route is not active
3. The request has timed out. The timeout can be increased by changing the timeout property on the connection information for the MQIPT.

**MQCAI100 This script is used to start the Administration Client for {0}. Specifying a SOCKS proxy will allow the Administrator Client to talk to an MQIPT through a firewall.**

**Explanation:** Online help information for mqiptGui script.

**MQCAI101 Format of command is :**

**Explanation:** Online help information for mqiptGui script.

**MQCAI102 mqiptGui {socks_host {socks_port}}**

**Explanation:** Online help information for mqiptGui script.

**MQCAI103 socks_host - host name of SOCKS proxy (optional)**

**Explanation:** Online help information for mqiptGui script.

**MQCAI104 socks_port - SOCKS proxy port address (optional - default 1080)**

**Explanation:** Online help information for mqiptGui script.

**MQCPA100 This script is used to stop or refresh {0}.**

**Explanation:** Online help information for the mqiptAdmin script.

**MQCPA101 (-stop | -refresh | -status) {hostname {port}}**

**Explanation:** Online help information for the mqiptAdmin script.

**MQCPA102 hostname - host name running MQIPT (default localhost)**

**Explanation:** Online help information for the mqiptAdmin script.

**MQCPA103 port - port address MQIPT is listening on for commands (default 1881)**

**Explanation:** Online help information for the mqiptAdmin script.

**MQCPA104 Command complete from MQIPT server at {0}**

**Explanation:** Command sent from IPTAdmin has been accepted and run by IPTController

**MQCPE001 Directory does not exist or is not a directory**

**Explanation:** At MQIPT initialization, a required directory could not be found. This message refers to a directory specified either in the MQIPT configuration file mqipt.conf or in the MQIPT command line startup options on the default directory.

**User response:** Specify the correct directory and retry the command.

**MQCPE004 Route startup failed on port {0}**

**Explanation:** It was not possible to start the route with the specified ListenerPort number.

**User response:** An I/O error occurred during route startup. Check for other adjacent error messages and log records to provide further explanation of the problem.

**MQCPE005 The configuration file {0} could not be found**

**Explanation:** The MQIPT configuration file mqipt.conf could not be found in the specified directory

**User response:** Specify the correct directory and retry the command.

**MQCPE006 The number of routes has exceeded {0}. MQIPT will start but this configuration is unsupported**

**Explanation:** Your configuration has exceeded the maximum supported number of routes for one instance of MQIPT. Operation will not be halted but the system might become unstable or overloaded as a result. Configurations that exceed the stated maximum number of routes will not be supported.

**User response:** Consider starting additional instances of MQIPT with fewer routes per instance.

**MQCPE007 Route not restarted on listener port {0}**

**Explanation:** On a REFRESH operation, the route that was operating on the specified ListenerPort was not restarted with the new configuration.

**User response:** Check for other adjacent error messages for further explanation of the problem.

**MQCPE008 Duplicate route defined for listener port {0}**

**Explanation:** More than one route has been defined with the same ListenerPort value.

**User response:** Remove the duplicate route from the configuration file and retry the command.

**MQCPE009 Log directory {0} is not valid**

**Explanation:** The log path shown in the text either does not exist or is not accessible at the time.

**User response:** Check the directory exists and is accessible by MQIPT.

**MQCPE010 Listener or command port number {0} is not valid**

**Explanation:** The port address supplied for the command port or listener port parameter is invalid.

**User response:** Specify a valid port address that is free for use. For guidance on use of port addresses in your network, consult your network administrator.

**MQCPE012 The value {0} is not valid for the property {1}**

**Explanation:** An invalid property value has been specified.

**User response:** Refer to this User Guide for full details of the valid values for each property.

**MQCPE013 ListenerPort property was not found in route {0}**

**Explanation:** MQIPT has detected a route in the configuration file that does not contain a ListenerPort property. The ListenerPort property is the primary and unique identifier for each route, and is therefore mandatory.

**User response:** Specify a valid ListenerPort property for the given route.

**MQCPE014 ListenerPort property value {0} is not valid**

**Explanation:** An invalid port address has been specified for the ListenerPort property of a route.

**User response:** A port address must be in the range

1024-65535. Check each ListenerPort in the configuration file.

**MQCPE015 No text was found for message number {0}**

**Explanation:** An internal error has been encountered for which no description is available.

**User response:** The mqipt.properties file may have become corrupted and the specified message number could not be found. Check the following:

1. if you are using the MQIPT_PATH environment variable, ensure it is set correctly
2. look in the Readme file for a possible new message
3. the mqipt.properties file is in the com.ibm.mq.ipt.jar file
4. the message number is in the mqipt.properties file

**MQCPE016 The maximum number of connection threads is {0} but this is less than the minimum number of connection threads, which is {1}**

**Explanation:** Your configuration file has specified the minimum number of connection threads with a value greater than the maximum number of connection threads.

**User response:** This could be an error in a single route, a conflict between a global property and a route property, or a route property overriding the system default values. Refer to the earlier chapters of this User Guide for full details of the valid values and applicable defaults.

**MQCPE017 The exception {0} was thrown causing MQIPT to shut down**

**Explanation:** MQIPT has abnormally terminated and has been shut down. This may have occurred because of system environmental conditions or constraints, such as memory overflow.

**User response:** If the condition persists, contact IBM Software Support.

**MQCPE018 The ListenerPort property is blank - the route will not start**

**Explanation:** The ListenerPort number has been omitted in a route.

**User response:** Edit the configuration file and add a valid ListenerPort.

**MQCPE019 The stanza {0} was not found before the following : {1}**

**Explanation:** A sequence error has occurred in the configuration file.

**User response:** Edit the configuration file and make sure all [route] entries are after the [global] entry.

**MQCPE020 The new value for MaxConnectionThreads is {0}. This must be greater than the current value {1}**

**Explanation:** After the route has started, the MaxConnectionThread property can only be increased.

**User response:** Edit the configuration file and change the MaxConnectionThread property.

**MQCPE021 The Destination property was not supplied for route {0}**

**Explanation:** The Destination property is mandatory for a route, but was omitted in the route specified.

**User response:** Edit the configuration file and add a Destination property for the given route.

**MQCPE022 The CommandPort value {0} is outside the valid range 1 - 65535**

**Explanation:** The CommandPort property was outside the range 1-65535.

**User response:** Edit the configuration file and change the CommandPort property to a valid port address.

**MQCPE023 Request for shutdown from Administration Client {0} is ignored because it is disabled**

**Explanation:** An attempt to shut down the MQIPT remotely has failed because remote shutdown was not enabled in the configuration file.

**User response:** To enable remote shutdown of MQIPT, edit the configuration file and set the RemoteShutDown property to true.

**MQCPE024 The command received by the MQIPT controller has not been recognized**

**Explanation:** The MQIPT has received a command through its command port which it does not recognize.

**User response:** Check the mqipt.log file for the identity of the command.

**MQCPE025 Failed to connect to server on host {0}, port {1}**

**Explanation:** The line mode (non-GUI) Administration Client has failed to communicate with the MQIPT.

**User response:** Make sure the CommandPort property has been specified as {1} in the configuration file and MQIPT is running on host {0}.

**MQCPE026 No reply received from server on host {0}, port {1}**

**Explanation:** The line mode (non-GUI) Administration Client has connected with the MQIPT but has not received a reply.

**User response:** This indicates that either the request has timed-out or there is a problem with the MQIPT.

**MQCPE027 Reply from MQIPT not recognized**

**Explanation:** The line mode (non-GUI) Administration Client has received a reply from the MQIPT, which it does not recognize.

**User response:** Check the mqiptAdmin script is using the same version of the MQipt.jar file as MQIPT.

**MQCPE028 Invalid stanza detected : {0}**

**Explanation:** The stated unrecognized stanza has been found in the configuration file.

**User response:** Only [global] and [route] stanzas are valid in the configuration file.

**MQCPE029 Was not able to flush log output**

**Explanation:** Some messages might not have been written to the log because the communication buffer could not be flushed.

**User response:** Check there is MQIPT home directory disk has not become full and MQIPT still has access to the logs subdirectory.

**MQCPE030 {0} not found in CLASSPATH**

**Explanation:** The specified jar file was not found in the system environment CLASSPATH variable.

**User response:** Add the specified file to the system CLASSPATH.

**MQCPE031 {0} class not found**

**Explanation:** This message is generated when displaying the version number of MQIPT. The specified class could not be found in the MQIPT jar file or the system environment CLASSPATH variable has been corrupted.

**User response:** Check the specified class file is in the MQipt.jar file and the MQipt.jar file is in the system CLASSPATH.

**MQCPE033 Failed to send configuration file to Administration Client at {0}**

**Explanation:** An error occurred sending the configuration file to the Administration Client.

**User response:** Check the configuration file is in the MQIPT home directory and is not being shared by another process.

**MQCPE034 Administration Client at {0} did not supply the correct password**

**Explanation:** The AccessPW property in the configuration file did not match that provided by the Administration Client.

**User response:** Either change the AccessPW property in the configuration file or the saved password in the Administration Client.

**MQCPE035 Failed to start command listener on port {0}**

**Explanation:** An I/O error occurred starting the CommandPort listener on the specified port address.

**User response:** Check the port address used for the CommandPort property in the configuration file.

**MQCPE038 MQIPT has not started as expected**

**Explanation:** This message is generated by the mqipt fork process, which starts MQIPT as a system service.

**User response:** Check the error logs for more information. You can try increasing the sleep time IPTFork uses before it checks if MQIPT is running. Edit mqiptFork script and increase the parameter passed to IPTFork.

**MQCPE039 I/O error occurred running mqipt script**

**Explanation:** An error has occurred launching MQIPT from the fork process

**User response:** Check the system PATH environment variable contains the location of the JDK and the mqipt script has execute authority.

**MQCPE040 Interruption occurred running mqipt script**

**Explanation:** An error has occurred after launching MQIPT from the fork process.

**User response:** Check the error logs for more information. If the condition persists contact IBM Software Support.

**MQCPE042 There is a conflict with the following properties on route {0} :**

**Explanation:** Some properties can not be used with others. This message precedes the list of properties in conflict.

**User response:** Check the following error messages and take the appropriate action.

**MQCPE043 ....{0} and {1}**

**Explanation:** The following properties cannot both be set at the same time on the same route.

**User response:** Edit the configuration file and disable one of the specified properties on the given route.

**MQCPE044 {0} is only valid on the {1} operating system**

**Explanation:** Some features of MQIPT are only valid on certain platforms.

**User response:** Edit the configuration file and disable the specified property.

**MQCPE045 ....HTTP proxy or server name is missing**

**Explanation:** The HTTPProxy or HTTPServer property must be set if the HTTP property has been set to true.

**User response:** Edit the configuration file and define an HTTPProxy or HTTPServer for the given route.

**MQCPE048 Route startup failed on port {0}, exception was : {1}**

**Explanation:** It was not possible to start the route with the specified ListenerPort number.

**User response:** Check for other adjacent error messages and log records to provide further explanation of the problem.

**MQCPE049 Error starting or stopping the Java Security Manager \n{0}**

**Explanation:** An exception was thrown while trying to start or stop the Java Security Manager.

**User response:** The Java Security Manager has previously been enabled, but runtime permissions have not been enabled. Add a RuntimePermission for setSecurityManager to your local policy file. MQIPT must be restarted for the changes to take effect.

**MQCPE050 Security exception on port {0} from the Administration Client**

**Explanation:** A security exception was thrown while accepting a connection from the Administration Client.

**User response:** The Java Security Manager has previously been enabled, but permissions have not been granted for the host identified in the error message. To allow the host to connect to MQIPT, add a SocketPermission to accept/resolve connections on the port address of the CommandPort. The Java Security Manager must be restarted for any changes to take effect.

**MQCPE051 Security exception accepting a connection on route {0}**

**Explanation:** A security exception was thrown while accepting a connection on the specified route.

**User response:** The Java Security Manager has previously been enabled, but permissions have not been granted for the host identified in the error message. To allow the host to connect on this route, add a SocketPermission to accept/resolve connections for the ListenerPort. The Java Security Manager must be restarted for any changes to take effect.

**MQCPE052 Connection request on route {0} failed : {1}**

**Explanation:** This message is issued in the connection log to record a security exception for a connection request.

**User response:** The Java Security Manager has previously been enabled, but permissions have not been granted for the host identified in the error message. To allow the host to connect on this route, add a SocketPermission to accept/resolve connections for the ListenerPort. The Java Security Manager must be restarted for any changes to take effect.

**MQCPE053 Security exception making a connection to {0}({1})**

**Explanation:** A security exception was thrown while making a connection on the specified route.

**User response:** The Java Security Manager has previously been enabled, but permissions have not been granted for the target identified in the error message. To allow MQIPT to connect to the target on this route, add a SocketPermission to connect/resolve connections for the ListenerPort. The Java Security Manager must be restarted for any changes to take effect.

**MQCPE054 Connection request to {0}({1}) failed : {2}**

**Explanation:** This message is issued in the connection log to record a security exception for a connection request to a target host.

**User response:** The Java Security Manager has previously been enabled, but permissions have not been granted to make a connection to the target host identified in the error message. To allow MQIPT to connect to the target host, add a SocketPermission to connect/resolve connections for the ListenerPort. The Java Security Manager must be restarted for any changes to take effect.

**MQCPE055 ....Socks proxy name is missing**

**Explanation:** The SocksProxy property must be set if the SocksClient property has been set to true.

**User response:** Edit the configuration file and define a SocksProxy for the given route.

**MQCPE056 Conflict with route properties**

**Explanation:** Some properties cannot be used with others.

**User response:** Check the console messages for details of the error and take the appropriate action.

**MQCPE057 SSL protocol ({0}) was not recognized**

**Explanation:** The route has been put into SSL proxy mode and the initial data flow is not recognized.

**User response:** Make sure only SSL connections are being made to this route.

**MQCPE058 CONNECT request to {2}({3}) through {0}({1}) failed**

**Explanation:** An HTTP CONNECT request was sent to the HTTP proxy to create an SSL tunnel to the HTTP server. The HTTP proxy did not send back a "200 OK" response to this request.

**User response:** This can be caused by various problems. Enable tracing on the route and retry the connection. The trace file will show the real error.

**MQCPE059 There are no defined key ring files**

**Explanation:** An SSL client or server has been defined without specifying at least one key ring file.

**User response:** Use the SSLClientKeyRing and SSLClientCAKeyRing properties on the client side or SSLServerKeyRing and SSLServerCAKeyRing on the server side to define a key ring file and then restart the route.

**MQCPE060 Runtime error setting SSL client connect timeout to {0} seconds**

**Explanation:** An SSL runtime error has occurred on the client side setting the timeout value.

**User response:** Check the value specified in the SSLClientConnectTimeout property is valid. Running a trace on the given route will show further error information.

**MQCPE061 There are no enabled cipher suites**

**Explanation:** An SSL client or server connection has been started but MQIPT is unable to determine a valid cipher suite.

**User response:** Check there are valid certificates in the defined key ring file(s). The private and public keys used to generate the certificates and the encryption algorithms used must comply with the list of supported cipher suites, which can be found in the MQIPT book.

**MQCPE062 Runtime error setting SSL cipher suite {0}**

**Explanation:** An unsupported SSL cipher suite has been defined on the client or server side.

**User response:** Check the value specified in the SSLClientCipherSuites or SSLServerCipherSuites is valid and supported on this connection. Running a trace on the given route will show the list of enabled cipher suites. The MQIPT book contains a list of supported cipher suites.

**MQCPE063 File {0} already exists - use the replace option**

**Explanation:** The file name parameter specified for the mqiptPW script already exists.

**User response:** Either choose another file name or use the replace option.

**MQCPE064 Runtime error generating decryption keys :\n {0}**

**Explanation:** An error has occurred while generating cipher keys to decrypt the password used to open a key ring file.

**User response:** The runtime error listed in the message should be rectified and the command run again.

**MQCPE065 ....LDAP server name is missing**

**Explanation:** The LDAPServer1 or LDAPServer2 property must be set if the LDAP property has been set to true.

**User response:** Edit the configuration file and define an LDAPServer* for the given route.

**MQCPE066 ....LDAP password is missing for the LDAPServer{0}Password property**

**Explanation:** An LDAP userid has been specified without a password, for either the main or backup LDAP server.

**User response:** Edit the configuration file and define an LDAP password for the given route. The LDAPServer1Password property is for the main server and LDAPServer2Password property is for the backup server.

**MQCPE067 ....SSLClient or SSLServer missing for LDAP server**

**Explanation:** The SSLClient or SSLServer property must be set if the LDAP property has been set to true.

**User response:** Edit the configuration file and define an SSLClient or SSLServer for the given route.

**MQCPE068 ....Security exit name is missing**

**Explanation:** The SecurityExitName property must be set if the SecurityExit property has been set to true.

**User response:** Edit the configuration file and define a SecurityExitName for the given route.

**MQCPE071 Error writing to {0}**

**Explanation:** An error occurred while creating or updating the file containing the encryted password. The error message also contains the exception thrown.

**User response:** This error is generated from the the mqiptPW script. The error listed in the exception should be rectified and the command run again.

**MQCPE072 An unknown error occurred in security exit {0}**

**Explanation:** An error occurred in a user-defined security exit while validating a connection request.

**User response:** Enable tracing in the security exit and try the connection request again. The error will be recorded in the security exit trace file.

**MQCPE073 Security exit {0} timed out**

**Explanation:** A user-defined security exit timed out while validating a connection request.

**User response:** Increase the timeout period for the security exit and try the connection request again.

**MQCPE074 ....Certificate exit name is missing**

**Explanation:** The SSLExitName property must be set if the SSLClientExit or SSLServerExit property has been set to true.

**User response:** Edit the configuration file and define a SSLExitName for the given route.

**MQCPE075 ....SSLPlainConnections needs SSLServer or SSLProxyMode enabled**

**Explanation:** The SSLExitName property must be set if the SSLClientExit or SSLServerExit property has been set to true.

**User response:** Edit the configuration file and define a SSLExitName for the given route.

**MQCPE076 Route {0} property {1} contains unsupported CipherSuites. The following CipherSuites are unsupported: {2}**

**Explanation:** At least one unsupported CipherSuite was included in the SSLClientCipherSuites or SSLServerCipherSuites

**User response:** Edit the configuration file and remove the unsupported CipherSuite from the route configuration.

**MQCPE077 Route {0} property {1} specifies file location {2} which does not exist.**

**Explanation:** A route property refers to a file or directory which does not exist.

**User response:** Edit the configuration file and specify the correct location for the file or directory.

**MQCPE078 Route {0} property {1} specifies file location {2} which cannot be read.**

**Explanation:** A route property refers to a file cannot be read.

**User response:** Ensure that the file permissions allow MQIPT to read it.

**MQCPE079 Route {0} site certificate label {1} was not found in keyring file {2}.**

**Explanation:** A site certificate label was specified but it was not found in the keyring file.

**User response:** Ensure that correct site certificate label is specified and that the certificate exists in the appropriate keyring.

**MQCPE080 Unable to determine MQIPT installation directory. Set the MQIPT_PATH environment variable to the absolute path of the top-level MQIPT directory.**

**Explanation:** The MQIPT command was unable to determine the installation directory.

**User response:** Set the MQIPT_PATH environment variable to the absolute path of the top-level MQIPT directory.

**MQCPE081 Invalid MQIPT_PATH {0}. The directory does not exist or does not contain a valid MQIPT installation.**

**Explanation:** The MQIPT_PATH environment variable is set incorrectly. Either the directory does not exist or the directory is not an MQIPT installation.

**User response:** Check the MQIPT_PATH environment variable is set correctly and re-run the command.

**MQCPE082 Unable to install the MQIPT service because a service is already installed. Only one MQIPT service may be installed at a time.**

**Explanation:** The user attempted to install the MQIPT service, but an MQIPT service is already installed. Only one MQIPT service may be installed on the system at a time.

**User response:** Merge the required routes into the existing MQIPT service configuration, or remove the existing service and install the new service in its place.

**MQCPE083 Unable to remove the MQIPT service because the installed service was not installed by the current MQIPT installation. Run mqiptService from the MQIPT installation that installed the service.**

**Explanation:** The MQIPT service may only be removed using the MQIPT installation that originally installed it. This error occurs when you have multiple MQIPT installations on the system and you attempt to remove the MQIPT service using a different installation from the one that originally installed it.

**User response:** Run mqiptService -remove from the correct MQIPT installation.

**MQCPE084 The MQIPT service is not installed.**

**Explanation:** The user attempted to remove the MQIPT service but there is no MQIPT service installed.

**MQCPE085 Error refreshing the Java Security Manager policy\n{0}**

**Explanation:** An exception was thrown while trying to refresh the Java Security Manager policy.

**User response:** Investigate the cause of the error and ensure that the updated policy file has the correct syntax.

**MQCPE086 Security exit {0} for route {1} failed to initialize due to error {2}.**

**Explanation:** The security exit initialization method returned an unexpected error, which prevented the route from starting.

**User response:** Investigate the cause of the error and restart the route.

**MQCPE087 Security exit {0} for route {1} failed to load due to error {2}.**

**Explanation:** The security exit could not be loaded, which prevented the route from starting.

**User response:** Investigate the cause of the exit load error and restart the route.

**MQCPE088 Certificate exit {0} for route {1} failed to initialize due to error {2}.**

**Explanation:** The certificate exit initialization method returned an unexpected error, which prevented the route from starting.

**User response:** Investigate the cause of the error and restart the route.

**MQCPE089 Certificate exit {0} for route {1} failed to load due to error {2}.**

**Explanation:** The certificate exit could not be loaded, which prevented the route from starting.

**User response:** Investigate the cause of the exit load error and restart the route.

**MQCPE090 The security exit rejected the connection with return code {0} and error {1}.**

**Explanation:** The security exit rejected a connection to the route listener port.

**User response:** Investigate the error returned by the exit.

**MQCPE091 The SSLClient certificate exit rejected the connection with return code {0} and error {1}.**

**Explanation:** The SSLClient certificate exit rejected the remote server certificate.

**User response:** Investigate the error returned by the exit.

**MQCPE092 The SSLServer certificate exit rejected the connection with return code {0} and error {1}.**

**Explanation:** The SSLServer certificate exit rejected the remote client certificate.

**User response:** Investigate the error returned by the exit.

**MQCPE093 Global property {0} specifies file location {1} which does not exist.**

**Explanation:** A global property refers to a file or directory which does not exist.

**User response:** Edit the configuration file and specify the correct location for the file or directory.

**MQCPE094 Global property {0} specifies file location {1} which cannot be read.**

**Explanation:** A global property refers to a file cannot be read.

**User response:** Ensure that the file permissions allow MQIPT to read it.

**MQCPE095 The MQIPT installation directory {0} must not contain a space on this platform.**

**Explanation:** The MQIPT installation directory contains a space character, which is not supported on the IBM AIX, HP-UX, Linux or Solaris platforms.

**User response:** Rename the installation directory so that it does not contain a space.

**MQCPE096 Error enabling TCP keep alive**

**Explanation:** The TCP keep alive route property is set, but MQIPT was unable to enable TCP keep alive.

**User response:** Investigate the cause of the failure or disable TCP keep alive.

**MQCPI001 {0} starting**

**Explanation:** This MQIPT instance is beginning execution. Further initialization messages will follow.

**MQCPI002 {0} shutting down**

**Explanation:** MQIPT is going to shut down. This can result from a STOP command, or automatically if a configuration error prevents a successful startup or REFRESH action.

**MQCPI003 {0} shutdown complete**

**Explanation:** The shutdown process has completed. All MQIPT processes are now ended.

**MQCPI004 Reading configuration information from {0}**

**Explanation:** The MQIPT configuration file mqipt.conf is being read from the directory described in this message.

**MQCPI005 Listener port specified as not active - {0} -> {1}({2})**

**Explanation:** The route referred to in the message has been marked as inactive. No communication requests will be accepted on this route.

**MQCPI006 Route {0} is starting and will forward messages to :**

**Explanation:** A route has been started on the listener port shown in this message. This message is followed by other messages listing any properties associated with this route. Message MQCPI078 will be issued when the route is ready to accept connections.

**MQCPI007 Route {0} has been stopped**

**Explanation:** The route that was operating on the specified ListenerPort is being shut down. This action normally occurs when a REFRESH command is issued to MQIPT and the route configuration has been changed.

**MQCPI008 Listening for control commands on port {0}**

**Explanation:** This MQIPT instance is listening for control commands on the specified port.

**MQCPI009 Control command received: {0}**

**Explanation:** This message indicates that a control command has been received at the command port. Where applicable, details are included in the message.

**MQCPI010 Stopping command port on {0}**

**Explanation:** On a REFRESH operation, the command port is no longer in use in the new configuration. Commands will no longer be accepted at the specified port.

**MQCPI011 The path {0} will be used to store the log files**

**Explanation:** Logging output will be directed to the location described in this message, under the current configuration.

**User response:** This may change if the configuration is amended and a REFRESH operation is requested.

**MQCPI012 Changing the value of MinConnectionThreads has no effect after the route is started**

**Explanation:** The minimum number of connection threads is assigned at route startup and cannot be changed until MQIPT is restarted.

**MQCPI013 Connection from {0} to host {1} closed**

**Explanation:** This message is issued in the connection log to record connection activity.

**MQCPI014 Eyecatcher protocol ({0}) not recognized**

**Explanation:** This message is issued in the connection log to record connection activity.

**MQCPI015 Client access has been disabled on this route**

**Explanation:** This message is issued in the connection log to record connection activity.

**MQCPI016 Queue Manager access has been disabled on this route**

**Explanation:** This message is issued in the connection log to record connection activity.

**MQCPI017 A queue manager on {0} was connected to host {1}**

**Explanation:** This message is issued in the connection log to record connection activity.

**MQCPI018 A client on {0} was connected to host {1}**

**Explanation:** This message is issued in the connection log to record connection activity.

**MQCPI019 {0} routes have been created - this exceeds the maximum number of supported routes, which is {1}**

**Explanation:** The maximum number of supported routes has been exceeded.

**User response:** MQIPT will continue to operate, but you might want to create a second MQIPT instance and split the routes between the two.

**MQCPI020 The configuration file has been sent to Administration Client {0}**

**Explanation:** As a result of a request from the Administration Client, the configuration file has been sent.

**MQCPI021 Password checking has been enabled on the command port**

**Explanation:** This message shows that a password is required to access the command port.

**MQCPI022 Password checking has been disabled on the command port**

**Explanation:** This message shows that a password is not required to access the command port.

**MQCPI024 ....and HTTP proxy at {0}({1})**

**Explanation:** This message indicates that the outgoing connection for this route will be made using this HTTP proxy.

**MQCPI025 The refresh requested by Administration Client {0} has finished**

**Explanation:** As a result of receiving a REFRESH command, the MQIPT has reread its configuration file and restarted.

**MQCPI026 Administration Client {0} has requested shutdown**

**Explanation:** As a result of receiving a STOP command, the MQIPT is shutting down.

**MQCPI027 {0} sent to {1} on port {2}**

**Explanation:** This displays on the system console the command sent by the line mode (non-GUI) Administration Client to the designated MQIPT.

**MQCPI031 ......cipher suites {0}**

**Explanation:** This message lists the cipher suites in use for this route.

**MQCPI032 ......keyring file {0}**

**Explanation:** This message gives the file name of the key ring for this route.

**MQCPI033 ......client authentication set to {0}**

**Explanation:** This message defines whether an SSL server is requesting client authentication for this route.

**MQCPI034 ....{0}({1})**

**Explanation:** This message shows the destination and destination port address for this route.

**MQCPI035 ....using {0}**

**Explanation:** This message shows the protocol being used to the destination. It will either be IBM MQ protocol, HTTP tunneling or HTTP chunking.

**MQCPI036 ....SSL Client side enabled with properties :**

**Explanation:** This message shows that the route will be using SSL to send data to the destination host.

**MQCPI037 ....SSL Server side enabled with properties :**

**Explanation:** This message shows that the route will be using SSL to receive data from the sending host.

**MQCPI038 ......peer certificate uses {0}**

**Explanation:** This message lists the distinguished names used to control authentication of peer certificates.

**MQCPI039 ....and Socks proxy at {0}({1})**

**Explanation:** This message shows that the outgoing connection for this route will be made using this Socks proxy, which is defined when MQIPT is started from the command line.

**MQCPI040 Command port has been accessed by Administration Client {0}**

**Explanation:** This message is written to the system console and the MQIPT log file (if logging is enabled). The MQIPT has received a connection from the Administration Client.

**MQCPI041 ....will reply to Network Dispatcher advisor requests in {0} mode**

**Explanation:** This message is written to the system console when a route is started. Used to show which mode MQIPT will use to reply to the Network Dispatcher advisor. Valid options are "Normal" and "Replace" mode.

**MQCPI042 Maximum connections reached on route {0} - further requests will be blocked**

**Explanation:** This message is written to the system console when the maximum number of connections has been reached for the given route. Further requests will be blocked until a connection becomes free or the MaxConnectionThreads value is increased.

**MQCPI043 Connections on route {0} now unblocked**

**Explanation:** This message is written to the system console when the given route is unblocked for connection requests.

**MQCPI044 MQIPT has been launched from system startup**

**Explanation:** MQIPT has been started as a system service.

**MQCPI045 Launching MQIPT from system startup**

**Explanation:** MQIPT is going to be started as a system service.

**MQCPI046 Sleeping for {0} seconds while MQIPT is launched from system startup**

**Explanation:**

**MQCPI047 ......CA keyring file {0}**

**Explanation:** This message gives the file name of the CA key ring for this route.

**MQCPI048 The ping by Administration Client {0} has finished**

**Explanation:** Response message from the IPTController to Administration Client.

**MQCPI050 Adding entry to inittab to automatically start MQIPT at system startup**

**Explanation:** User has run the mqiptService script to start MQIPT as a system service.

**MQCPI051 Removing entry from inittab that automatically starts MQIPT at system startup**

**Explanation:** User has run the mqiptService script to remove MQIPT from starting as a system service.

**MQCPI052 ....Socks server side enabled**

**Explanation:** This route will act as a SOCKS server (proxy) and will accept connections from a socksified application.

**MQCPI053 Starting the Java Security Manager**

**Explanation:** The default Java Security Manager will be started as the SecurityManager property has been set to true.

**MQCPI054 Stopping the Java Security Manager**

**Explanation:** The default Java Security Manager will be stopped as the SecurityManager property has been set to false.

**MQCPI055 Setting the java.security.policy to {0}**

**Explanation:** The default Java Security Manager is about to be started and will use the supplied policy file.

**MQCPI057 ....trace level {0} enabled**

**Explanation:** This message is written to the system console when a route is started. Used to show the level of tracing enabled on this route.

**MQCPI058 ....and a URI name of {0}**

**Explanation:** This message is written to the system console when a route is started. Used to show the Uniform Resource Identifier name on this route.

**MQCPI059 ....servlet client enabled**

**Explanation:** This message is written to the system console when a route is started. This route will connect to the MQIPT servlet.

**MQCPI060 Installing files to automatically start MQIPT at system startup**

**Explanation:** User has run the mqiptService script to start MQIPT as a system service.

**MQCPI061 Removing files that automatically starts MQIPT at system startup**

**Explanation:** User has run the mqiptService script to remove MQIPT from starting as a system service.

**MQCPI064 ......no SSL authentication on this route**

**Explanation:** This message is written to the system console when a route is started and shows there is no SSL authentication is in use for this route, as an anonymous cipher suite has been specified.

**MQCPI066 ....and HTTP server at {0}({1})**

**Explanation:** This message indicates that the outgoing connection for this route will be made using this HTTP server.

**MQCPI069 ....binding to local address {0} when making new connections**

**Explanation:** This message shows the local IP address each new connection is bound to the destination address. This should only be used on a multihomed system.

**MQCPI070 ....using local port address range {0}-{1} when making new connections**

**Explanation:** This message shows the local port addresses that will be used for new connections. This will allow firewall administrators to restrict connections from MQIPT.

**MQCPI071 ......site certificate uses {0}**

**Explanation:** This message lists the distinguished names used to control selection of a site certificate.

**MQCPI072 ......and certificate label {0}**

**Explanation:** This message lists the label name used to control selection of a site certificate.

**MQCPI073 Updated file {0}**

**Explanation:** The file name specified for the mqiptPW script has been updated.

**MQCPI074 Created file {0}**

**Explanation:** The file name specified for the mqiptPW script has been created.

**MQCPI075 ....LDAP main server at {0}({1})**

**Explanation:** This message lists the name of the main LDAP server used for CRL support.

**MQCPI076 ....LDAP backup server at {0}({1})**

**Explanation:** This message lists the name of the backup LDAP server used for CRL support.

**MQCPI077 ....LDAP errors will be ignored**

**Explanation:** This message means that any errors received from LDAP will be ignored.

**MQCPI078 Route {0} ready for connection requests**

**Explanation:** This message is displayed when a route is ready to accept connection requests.

**MQCPI079 ....using security exit {0}**

**Explanation:** This message is written to the system console when a route is started. Used to show the fully qualified name of the security exit.

**MQCPI080 ......and timeout of {0} second(s)**

**Explanation:** This message is written to the system console when a route is started. Used to show the timeout value of the security or certificate exit.

**MQCPI081 Start message for IBM MQ Internet Pass-Thru**

**Explanation:** Start message for IBM MQ Internet Pass-Thru as a service

**MQCPI082 Stop message for IBM MQ Internet Pass-Thru**

**Explanation:** Stop message for IBM MQ Internet Pass-Thru as a service

**MQCPI083 ....refresh commands will not restart the route**

**Explanation:** This message indicates that when a refresh command has been issued the route will not be restarted.

**MQCPI084 ......CRL cache expiry timeout is {0} hour(s)**

**Explanation:** This console message displays how long a CRL (or ARL) will remain in the MQIPT cache.

**MQCPI085 ....CRLs will be saved in the key ring file(s)**

**Explanation:** This console message means that any CRLs (or ARLs) retrieved from an LDAP server will be saved in the key ring file, attached to the associated CA certificate.

**MQCPI086 ......timeout of {0} second(s)**

**Explanation:** This message is written to the system console when a route is started. Used to show the timeout value for connecting to the LDAP server.

**MQCPI087 ......userid is {0}**

**Explanation:** This message is written to the system console when a route is started. Used to show the userid name to connect to the LDAP server.

**MQCPI088 ....buffer size {0}**

**Explanation:** This message is written to the system console when a route is started. Used to show the size of buffers being used, but only if not the value of 65535. This value will only be used if greater than the default 65535.

**MQCPI089 ....use cookies {0}**

**Explanation:** This message is written to the system console when a route is started. Used to show which cookies will be passed to the MQIPT servlet, as returned in the previous HTTP response.

**MQCPI090 ......search baseDN uses {0}**

**Explanation:** This message is written to the system console when a route is started. Used to show the LDAP baseDN key names to retrieve CRLs (and ARLs).

**MQCPI091 ....allow plain connections**

**Explanation:** This message is written to the system console when a route is started. Used to show plain connections are allowed when acting as an SSL server or running in SSL proxy mode.

**MQCPI092 ....socket timeout {0} ms**

**Explanation:** This message shows the socket timeout value (in milliseconds)

**MQCPI100 This script is used to start {0}**

**Explanation:** Online help message from mqipt script.

**MQCPI101 Format of command is :**

**Explanation:** Online help message from MQIPT script.

**MQCPI102 mqipt {dir_name}**

**Explanation:** Online help message from mqipt script.

**MQCPI103 dir_name - directory containing mqipt.conf**

**Explanation:** Online help message from mqipt script.

**MQCPI106 This script is used to display the current version number**

**Explanation:** Online help message from mqiptVersion script.

**MQCPI107 mqiptVersion {-v}**

**Explanation:** Online help message from mqiptVersion script.

**MQCPI108 where -v will also display the build timestamp**

**Explanation:** Online help message from mqiptVersion script.

**MQCPI109 This script is used to start {0}, from system startup, in another JVM and is only used in mqipt.ske. Use the mqipt script to start MQIPT from the command line.**

**Explanation:** Online help message from mqiptFork script.

**MQCPI110 This class is used to display a simple NLS message on the console.**

**Explanation:** Online help message from IPTMessages class.

**MQCPI111 java com.ibm.mq.ipt.IPTMessages (message_id1) {message_id2} {message_id...}**

**Explanation:** Online help message from IPTMessages class.

**MQCPI112 where message_id matches a key in the file mqipt.properties**

**Explanation:** Online help message from IPTMessages class.

**MQCPI113 This script is used to manage MQIPT as a system service.**

**Explanation:** Online help message from mqiptService script.

**MQCPI114 mqiptService (-install | -remove )**

**Explanation:** Online help message from mqiptService script.

**MQCPI115 -install will install files to start MQIPT automatically at system startup**

**Explanation:** Online help message from mqiptService script.

**MQCPI116 -remove will remove files that start MQIPT automatically at system startup**

**Explanation:** Online help message from mqiptService script.

**MQCPI121 Use this script to encrypt a password and store it in a file**

**Explanation:** Online help message from mqiptPW script.

**MQCPI122 mqiptPW password file_name { -replace }**

**Explanation:** Online help message from mqiptPW script.

**MQCPI123 password - password used to open a key ring file**

**Explanation:** Online help message from mqiptPW script.

**MQCPI124 file_name - encrypted password will be stored in this file**

**Explanation:** Online help message from mqiptPW script.

**MQCPI125 replace option must be used to update an existing file**

**Explanation:** Online help message from mqiptPW script.

**MQCPI126 mqipt (-start | -stop )**

**Explanation:** Online help message from mqipt script.

**MQCPI127 ....in full duplex mode**

**Explanation:** This message shows the HTTP protocol being used to the destination is working in full duplex mode.

**MQCPI128 ....in half duplex mode**

**Explanation:** This message shows the HTTP protocol being used to the destination is working in half duplex mode.

**MQCPI129 ......using certificate exit {0}**

**Explanation:** This message is written to the system console when a route is started. Used to show the fully qualified name of the certificate exit.

**MQCPI130 Connection to caller closed due to connection failure to destination**

**Explanation:** This message is written to the connection log for the closed connection to the caller, when MQIPT failed to connect to the target destination.

**User response:** See previous connection failure for reason of closure.

**MQCPI131 ......and certificate exit data "{0}"**

**Explanation:** This message is written to the system console when a route is started. Used to show the data for the certificate exit.

**MQCPI132 ....listening on local address {0}**

**Explanation:** This message shows the local IP address the route is listening on. This should only be used on a multihomed system.

**MQCPI133 This script starts the iKeyman certificate management utility.**

**Explanation:** This message introduces the usage statement for the mqiptKeyman command used to start the iKeyman certificate management utility.

**MQCPI134 mqiptKeyman**

**Explanation:** This message shows the usage statement for the mqiptKeyman command used to start the iKeyman certificate management utility.

**MQCPI135 This script runs the iKeycmd certificate management utility.**

**Explanation:** This message introduces the usage statement for the mqiptKeyman command used to start the iKeyman certificate management utility.

**MQCPI136 mqiptKeycmd {object} [{action} ...]**

**Explanation:** This message shows the usage statement for the mqiptKeyman command used to start the iKeyman certificate management utility.

**MQCPI137 mqiptIcons {-install | -remove} InstallationName**

**Explanation:** This message shows the usage statement for the mqiptIcons command used to install or remove MQIPT icons from the Windows Start menu.

**MQCPI138 The Java Security Manager policy has been refreshed.**

**Explanation:** The Java Security Manager is still enabled and the policy has been re-read. Any changes to the security policy will now take effect.

**MQCPI139 ......secure socket protocols {0}**

**Explanation:** This message lists the secure socket protocol versions enabled for this route.

**MQCPI140 ....TCP keep alive enabled**

**Explanation:** This message shows that TCP keep alive parameter has been enabled

**MQCPW001 CRL expired for {0}**

**Explanation:** This message is displayed when a CRL (or ARL) is retrieved from an LDAP server.

**User response:** Update the specified CRL in the LDAP server.

**MQCPW003 ....Expired CRLs will be ignored**

**Explanation:** This console message means that any expired CRLs (or ARLs) will be ignored and the connection request may be allowed.

**MQCPW004 ......SSLServerAskClientAuth is disabled, certificate exit might not be called**

**Explanation:** This console message is displayed at startup to show a conflict with the SSLServerExit and SSLServerAskClientAuth properties.

**User response:** With SSLServerAskClientAuth disabled, the SSL client is not required to send an SSL certificate, so the certificate exit might not be called.

**MQCPW005 Route {0} {1} keyring file {2} certificate {3} serial number {4} is not yet valid. The certificate cannot be used before {5}.**

**Explanation:** This console message is displayed at route startup if one of the keyring files contains a certificate which is not yet valid because its Not Before date is in the future.

**User response:** Check that the system clock is set correctly. If your organization operates its own CA, check the system clock on the CA system.

**MQCPW006 Route {0} {1} keyring file {2} certificate {3} serial number {4} has expired. The certificate cannot be used after {5}.**

**Explanation:** This console message is displayed at route startup if one of the keyring files contains a certificate which has expired.

**User response:** Check that the system clock is set correctly. If the clock is set correctly, obtain a replacement certificate.

# Index

## A

accessibility information 5
AccessPW property 124
Active property 126
administering MQIPT 115
administering MQIPT by using the command line 118
Administration Client 115
  administering MQIPT 116
  connection information 115
  file menu options 117
  inheritance of properties 116
  starting 115
AIX
  uninstalling MQIPT version 2.0 61
automatically starting MQIPT
  problems 145

## B

backing up key files 142

## C

certificate exit
  com.ibm.mq.ipt.exit.CertificateExit class 40
  com.ibm.mq.ipt.exit.CertificateExitResponse class 42
  com.ibm.mq.ipt.exit.IPTCertificate class 42
  com.ibm.mq.ipt.exit.IPTTrace class 43
  overview 40
certificate exit return codes 45
certificate related technologies 35
channel concentrator, MQIPT as a 1
chunking, HTTP 21
CipherSuites 25
client/server channels 17
ClientAccess property 126
cluster sender/receiver channels 17
clustering 23
command line commands 118
command port 58
CommandPort property 124
common problems 145
configuration
  default configuration file 120
  file protection 57
  property reference information 124, 125
  reference information 119
  summary of properties 120
  using the Administration Client 115
  using the command line 118
configuration file 11, 115
connection logs 58
connection scenarios 16
connection threads
  performance tuning 143
ConnectionLog property 125
control, port numbers 57
cryptographic algorithms 25

## D

denial of service attacks 57
Destination property 126
destination queue managers, access to 11
DestinationPort property 126
DMZ, MQIPT with 2

## E

encrypting key-ring password for iKeyman 37
encryption 3
end-to-end connectivity problems 149
example configurations 1, 69
  allocating port addresses 93
  Apache rewrite 101
  certificate exit to authenticate an SSL/TLS server 111
  configuring a SOCKS client 88
  configuring a SOCKS proxy 87
  configuring access control 85
  configuring MQIPT clustering support 90
  creating a key-ring file 72
  creating test certificates 75
  dynamic one route exit 108
  HTTP proxy configuration 83
  HTTP tunneling 81
  installation verification test 70
  routing security exit 105
  security exit 104
  SSL/TLS client authentication 79
  SSL/TLS proxy mode 97, 99
  SSL/TLS server authentication 77
  using an LDAP server 94
execution trace facility 150

## F

failure conditions 18
fault finding 145
features 21
FFST reports 145

## G

generic
  uninstalling MQIPT version 2.0 62
getting started with MQIPT 69

## H

handshake 31
heartbeat mechanism 21
high availability 19
HP-UX
  uninstalling MQIPT version 2.0 61
HTTP property 126
HTTP support 21
HTTP tunneling, HTTP with 3
HTTPProxy property 126
HTTPProxyPort property 126
HTTPS 22
HTTPS property 127
HTTPServer property 126
HTTPServerPort property 127

## I

idle timeout
  performance tuning 143
IdleTimeout property 127
IgnoreExpiredCRLs property 127
iKeyman 37, 38
  command line interface 37
  command-line interface 38
  encrypting key-ring password 37
  graphical user interface 37, 38
  key-ring file format 37
iKeyman and iKeycmd, tracing errors 150
inheritance of properties 116
installation verification test 70
installing 59
introduction 1

## J

Java Security Manager 49
JRE diagnostic options 149

## K

key-ring file
  selecting certificates 33
key-ring file format for iKeyman 37
keyring file
  encrypting a password 33
keyring password 33

## L

LDAP and CRLs 45
LDAP property 127
LDAPCacheTimeout property 127
LDAPIgnoreErrors property 127
LDAPSaveCRLs property 128
LDAPServer1 property 128
LDAPServer1Password property 128
LDAPServer1Port property 128

LDAPServer1Timeout property 128
LDAPServer1Userid property 128
LDAPServer2 property 128
LDAPServer2Password property 128
LDAPServer2Port property 128
LDAPServer2Timeout property 129
LDAPServer2Userid property 128
Linux
  service control program 148
  uninstalling MQIPT version 2.0 61
ListenerAddress property 129
ListenerPort property 129
LocalAddress property 129

## M

maintenance 142
MaxConnectionThreads property 129
MaxLogFileSize property 125
messages 153
messages, safety of 18
migrating from an earlier MQIPT 62
MinConnectionThreads property 129
MQIPT and SSL/TLS 32
mqipt.conf 115
multi-instance queue managers 19
multihomed systems 57
multiple queue managers 11

## N

Name property 129
NDAdvisor property 129
NDAdvisorReplaceMode property 130
Network Dispatcher 47
normal termination 18

## O

other security considerations 57
OutgoingPort property 130

## P

performance tuning 143
  connection threads 143
  idle timeout 143
port 57
port number control 57
prerequisites 4
privacy 5
problem determination 145
problems
  common 145
  end-to-end connectivity 149
properties
  global section 124
  route section 125
  summary 120
protocol forwarder, MQIPT as 11

## Q

QMgrAccess property 130

## R

refresh line mode command 119
RemoteShutDown property 125
reporting problems 150
requester/sender channels 18
requester/server channels 18
return codes
  certificate exit 45
  security exit 56
RouteRestart property 130

## S

safety of messages 18
security considerations, other 57
security exit
  com.ibm.mq.ipt.SecurityExit class 53
  com.ibm.mq.ipt.SecurityExitResponse class 55
  overview 52
  tracing 56
security exit return codes 56
SecurityExit property 130
SecurityExitName property 130
SecurityExitPath property 130
SecurityExitTimeout property 130
SecurityManager property 125
SecurityManagerPolicy property 125
sender/receiver channels 18
server/receiver channels 18
server/requester channels 18
service control program
  UNIX and Linux 148
  Windows 147
SOCKS support 22
SocksClient property 130
SocksProxyHost property 131
SocksProxyPort property 131
SocksServer property 131
Solaris
  uninstalling MQIPT version 2.0 61
SSL support
  handshake 31
SSL/TLS overview 24
SSL/TLS support 24
  error messages 35
  example 3
  LDAP and CRLs 45
  MQIPT and SSL/TLS 32
  testing 34
  trust settings 33
SSLClient property 131
SSLClientCAKeyRing property 131
SSLClientCAKeyRingPW property 131
SSLClientCipherSuites property 132
SSLClientConnectTimeout property 132
SSLClientDN_C property 132
SSLClientDN_CN property 132
SSLClientDN_DC property 132
SSLClientDN_DNQ property 132
SSLClientDN_L property 132
SSLClientDN_O property 133
SSLClientDN_OU property 133
SSLClientDN_PC property 133
SSLClientDN_ST property 133
SSLClientDN_Street property 133
SSLClientDN_T property 133
SSLClientDN_UID property 133
SSLClientExit property 134
SSLClientKeyRing property 134
SSLClientKeyRingPW property 134
SSLClientProtocols property 134
SSLClientSiteDN_C property 134
SSLClientSiteDN_CN property 134
SSLClientSiteDN_DC property 135
SSLClientSiteDN_DNQ property 135
SSLClientSiteDN_L property 135
SSLClientSiteDN_O property 135
SSLClientSiteDN_OU property 135
SSLClientSiteDN_PC property 135
SSLClientSiteDN_ST property 135
SSLClientSiteDN_Street property 136
SSLClientSiteDN_T property 136
SSLClientSiteDN_UID property 136
SSLClientSiteLabel property 136
SSLExitData property 136
SSLExitName property 136
SSLExitPath property 136
SSLExitTimeout property 136
SSLPlainConnections property 136
SSLProxyMode property 137
SSLServer property 137
SSLServerAskClientAuth property 137
SSLServerCAKeyRing property 137
SSLServerCAKeyRingPW property 137
SSLServerCipherSuites property 137
SSLServerDN_C property 137
SSLServerDN_CN property 138
SSLServerDN_DC property 138
SSLServerDN_DNQ property 138
SSLServerDN_L property 138
SSLServerDN_O property 138
SSLServerDN_OU property 138
SSLServerDN_PC property 139
SSLServerDN_ST property 139
SSLServerDN_Street property 139
SSLServerDN_T property 139
SSLServerDN_UID property 139
SSLServerExit property 139
SSLServerKeyRing property 139
SSLServerKeyRingPW property 139
SSLServerProtocols property 140
SSLServerSiteDN_C property 140
SSLServerSiteDN_CN property 140
SSLServerSiteDN_DC property 140
SSLServerSiteDN_DNQ property 140
SSLServerSiteDN_L property 141
SSLServerSiteDN_O property 141
SSLServerSiteDN_OU property 141
SSLServerSiteDN_PC property 141
SSLServerSiteDN_ST property 141
SSLServerSiteDN_Street property 141
SSLServerSiteDN_T property 141
SSLServerSiteDN_UID property 141
SSLServerSiteLabel property 142
starting MQIPT
  automatically 12
  from command line 12
stop line mode command 119
stopping MQIPT 13
support 145

## T

TCP/IP and MQIPT 11
TCPKeepAlive property 142
termination 18
thread pool management 143
topology of MQIPT 13
Trace property 125, 142
tracing errors
  iKeyman and iKeycmd 150
  MQIPT 150
troubleshooting 145
trust settings 33
tunnelling, HTTP 21

## U

uninstalling MQIPT version 2.0
  AIX 61
  generic UNIX 62
  HP-UX 61
  Linux 61
  Solaris 61
  Windows 62
uninstalling MQIPT version 2.1 62
UNIX
  service control program 148
upgrading MQIPT 66
UriName property 142
uses of MQIPT 1

## W

what's new in version 2.1 7
Windows
  service control program 147
  uninstalling MQIPT version 2.0 62

# Notices

This information was developed for products and services offered in the U.S.A.

IBM may not offer the products, services, or features discussed in this document in other countries. Consult your local IBM representative for information on the products and services currently available in your area. Any reference to an IBM product, program, or service is not intended to state or imply that only that IBM product, program, or service may be used. Any functionally equivalent product, program, or service that does not infringe any IBM intellectual property right may be used instead. However, it is the user's responsibility to evaluate and verify the operation of any non-IBM product, program, or service.

IBM may have patents or pending patent applications covering subject matter described in this document. The furnishing of this document does not grant you any license to these patents. You can send license inquiries, in writing, to:

IBM Director of Licensing
IBM Corporation
North Castle Drive
Armonk, NY 10504-1785
U.S.A.

For license inquiries regarding double-byte (DBCS) information, contact the IBM Intellectual Property Department in your country or send inquiries, in writing, to:

Intellectual Property Licensing
Legal and Intellectual Property Law
IBM Japan, Ltd.
19-21, Nihonbashi-Hakozakicho, Chuo-ku
Tokyo 103-8510, Japan

**The following paragraph does not apply to the United Kingdom or any other country where such provisions are inconsistent with local law:** INTERNATIONAL BUSINESS MACHINES CORPORATION PROVIDES THIS PUBLICATION "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. Some states do not allow disclaimer of express or implied warranties in certain transactions, therefore, this statement may not apply to you.

This information could include technical inaccuracies or typographical errors. Changes are periodically made to the information herein; these changes will be incorporated in new editions of the publication. IBM may make improvements and/or changes in the product(s) and/or the program(s) described in this publication at any time without notice.

Any references in this information to non-IBM Web sites are provided for convenience only and do not in any manner serve as an endorsement of those Web sites. The materials at those Web sites are not part of the materials for this IBM product and use of those Web sites is at your own risk.

## Programming interface information

Programming interface information, if provided, is intended to help you create application software for use with this program.

This book contains information on intended programming interfaces that allow the customer to write programs to obtain the services of WebSphere MQ.

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

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

## Trademarks

# Sending your comments to IBM

We appreciate your input on this publication. Feel free to comment on the clarity, accuracy, and completeness of the information or give us any other feedback that you might have.

Use one of the following methods to send us your comments:

- Send an email to ibmkc@us.ibm.com
- Use the form on the web here: www.ibm.com/software/data/rcf/

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

Include the following information:

- Your name and address
- Your email address
- Your telephone or fax number
- The publication title and order number
- The topic and page number related to your comment
- The text of your comment

IBM or any other organizations will only use the personal information that you supply to contact you about the issues that you submit.

Thank you for your participation.

SC34-2920-00