WebSphere MQ internet pass-thru
Version 1.3

SC34-6100-01

Note!

Before using this information and the product it supports, be sure to read the general information under "Notices", on page 167.

Fourth Edition (March 2003)

This edition applies to Version 1.3 of WebSphere MQ internet pass-thru (program number 5639-L92) and to all subsequent releases and modifications until otherwise indicated in new editions.

Contents

Figures

Preface

What is internet pass-thru?

WebSphere MQ internet pass-thru was formerly known as MQSeries internet pass-thru. WebSphere MQ is the name by which MQSeries will be known from now on in this book. Note that not all MQSeries manuals will change name to WebSphere MQ straight away, be aware that there will be references to both MQSeries and WebSphere MQ for some time.

IBM® WebSphere MQ internet pass-thru:

- Is a WebSphere MQ base product extension that can be used to implement messaging solutions between remote sites across the Internet
- Makes the passage of WebSphere MQ channel protocols into and out of a firewall simpler and more manageable by tunneling the protocols inside HTTP or acting as a proxy
- Operates as a standalone service that can receive and forward WebSphere MQ message flows. The system on which it runs does not have to host a WebSphere MQ queue manager
- Helps you to provide business-to-business transactions using WebSphere MQ
- Enables existing, unchanged WebSphere MQ applications to be used through a firewall
- Provides a single point of control over access to multiple queue managers
- Allows encryption of all data
- Logs all connection attempts

In this book, WebSphere MQ internet pass-thru is often termed “MQIPT” for convenience.

Who this book is for

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

What you need to know to understand this document

You need a good understanding of:

- The administration of WebSphere MQ queue managers and message channels, as described in *WebSphere MQ System Administration Guide* and *WebSphere MQ Intercommunication*
- The way that firewalls are implemented
- Internet protocol routing/networking
- The IBM Network Dispatcher for load balancing and enhanced availability
- IBM WebSphere® Application Server

Prerequisites

This release of internet pass-thru runs on these platforms:

- Windows NT® V4.0, with Service pack 6
- Windows® 2000
- Windows XP

- Sun Solaris
- AIX® V5.1
- HP-UX 11
- Linux

The J2SE V1.4.0 runtime (JRE) is required for the MQIPT server. The SDK, V1.4.0, is required to create a security exit.

The only supported network protocol is TCP/IP.

The Administration Client help requires a Netscape browser.

Accessibility information

The Administration Client 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 cursor keys in the standard manner. The equivalent to pressing buttons can be achieved by first selecting the button and then pressing the enter key.

Menu options can be reached either by combinations of tab and cursor keys or by using the accelerator keys, which are available for all the options. For example, the GUI can be closed by selecting first alt-f, then alt-q (File->Quit). Once a menu item has been reached, it can be activated by using the enter key.

You can navigate around the tree by using the cursor keys. In particular, the right and left cursor keys can be used to open or close an MQIPT node, allowing the routes to be either shown or hidden.

Selected checkboxes can have their states changed by using the space key. Fields can be selected for editing by using the enter key.

Look and feel

Ideally the GUI should adopt the look and feel of the environment. As this is not always possible, you may 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:

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

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

Summary of changes

This section describes changes in this edition of WebSphere MQ internet pass-thru. Changes since the previous edition are marked by vertical lines to the left of the changes.

Changes for this edition (SC34-6100-01)

Enhancements to this version of WebSphere MQ internet pass-thru include:

- A security exit to control client connection requests
- LDAP support for CRLs and ARLs
- Encryption of key ring passwords
- Certificate selection from a key ring
- New AES cipher suites
- Generic UNIX® disk image
- Control of route restart action
- AIX and HP-UX platforms now support Java™ 1.4

Changes for the third edition (SC34-6100-00)

The enhancements in this version of WebSphere MQ internet pass-thru include:

- Control outgoing port address allocation
- Example configurations
- Improved SSL tracing
- Java Security Manager
- KeyMan utility for managing SSL certificates and key ring files
- Linux support, including Quality of Service for WebSphere MQ messages
- NLS install image available on Windows platforms
- Property names are now case insensitive
- Servlet version
- Socks client and server support
- SSL proxy mode
- Support multihomed system
- Traffic light status for the Administration Client
- WebSphere MQ cluster support

Changes for the second edition

The enhancements in this version of WebSphere MQ internet pass-thru include:

- The addition of AIX, HP-UX, and Windows 2000 as platforms for MQIPT
- The addition of HTTP proxy support
- The addition of Secure Socket Layer (SSL) support
- The ability of MQIPT to communicate with another external MQIPT or MQSeries® server through a SOCKS proxy
- The use of an Administration Client GUI to make administration of one or more MQIPTs easier

- The addition of support for the IBM Network Dispatcher
- Minor improvements to tracing
- Minor improvements to the mqiptAdmin command

Chapter 1. Introduction to WebSphere MQ internet pass-thru

WebSphere MQ internet pass-thru is an extension to the base WebSphere MQ product. MQIPT runs as a standalone service that can receive and forward WebSphere MQ message flows, either between two WebSphere MQ queue managers or between a WebSphere MQ client and a WebSphere MQ queue manager. MQIPT enables this connection when the client and server are not on the same physical network.

One or more MQIPTs can be placed in the communication path between two WebSphere MQ queue managers, or between a WebSphere MQ client and a WebSphere MQ queue manager. The MQIPTs allow the two WebSphere 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 WebSphere MQ messages, WebSphere MQ messages tunneled inside HTTP, or encrypted using Secure Sockets Layer (SSL). It can handle multiple concurrent connections.

The WebSphere 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”.

The anticipated uses of MQIPT are:

- MQIPT can be used as a channel concentrator, so that channels from or to multiple separate hosts can appear to a firewall as if they are all from or to the MQIPT host. This makes it easier to define and manage firewall filtering rules.

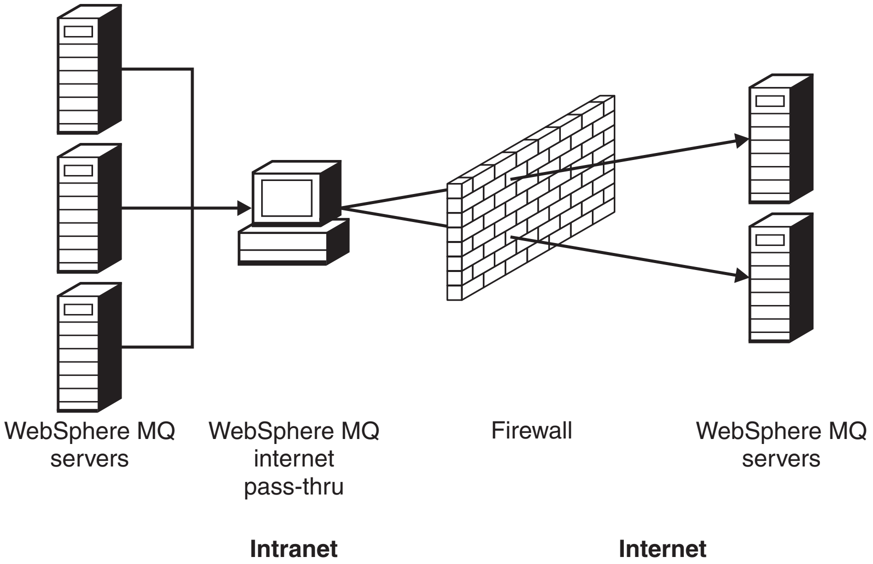


Figure 1. Example of MQIPT as a channel concentrator

- If MQIPT is placed in the firewall's "demilitarized zone" (DMZ), on a machine with a known and trusted internet protocol (IP) address, MQIPT can be used to listen for incoming WebSphere MQ channel connections which it can then forward to the trusted intranet; the inner firewall must allow this trusted machine to make inbound connections. In this configuration, MQIPT prevents external requests for access from seeing the true IP addresses of the machines in the trusted intranet. Thus, MQIPT provides a single point of access.

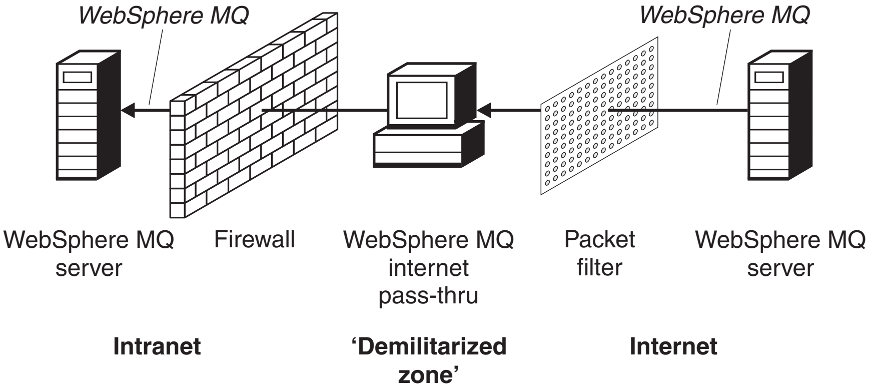


Figure 2. Example of MQIPT with a "demilitarized zone"

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

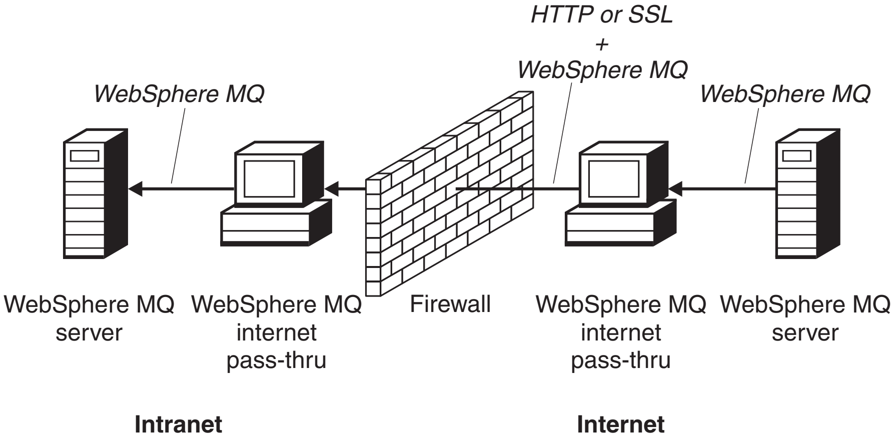


Figure 3. Example of MQIPT and HTTP tunneling

- Similarly, requests can be encrypted before transmission through firewalls. The first MQIPT encrypts the data and the second decrypts it using SSL before sending it to the destination queue manager.

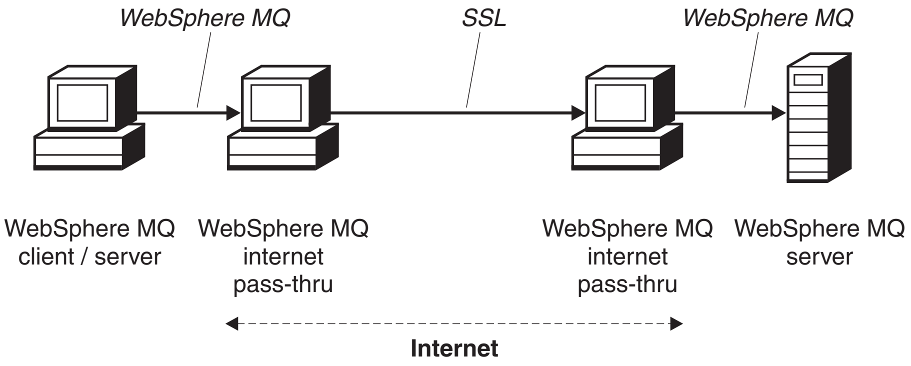

WebSphere MQ
SSL
WebSphere MQ
WebSphere MQ client / server
WebSphere MQ internet pass-thru
WebSphere MQ internet pass-thru
WebSphere MQ server
Internet

Figure 4. Example of MQIPT and SSL

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 log and trace records.

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

MQIPT can be used in conjunction with WebSphere MQ Publish/Subscribe or the WebSphere MQ Integrator message broker.

Figure 5 on page 4 shows all the possible configurations for MQIPTs in a WebSphere MQ topology. In the figure, note that the HTTP proxy, SOCKS proxy, and MQIPT machines beyond the firewall on the “Outbound connections” side represent the possibility of multiple machines chained together on the internet. For example, an MQIPT machine could communicate through one or more SOCKS or HTTP proxy machines, or further MQIPT machines, before reaching its target.

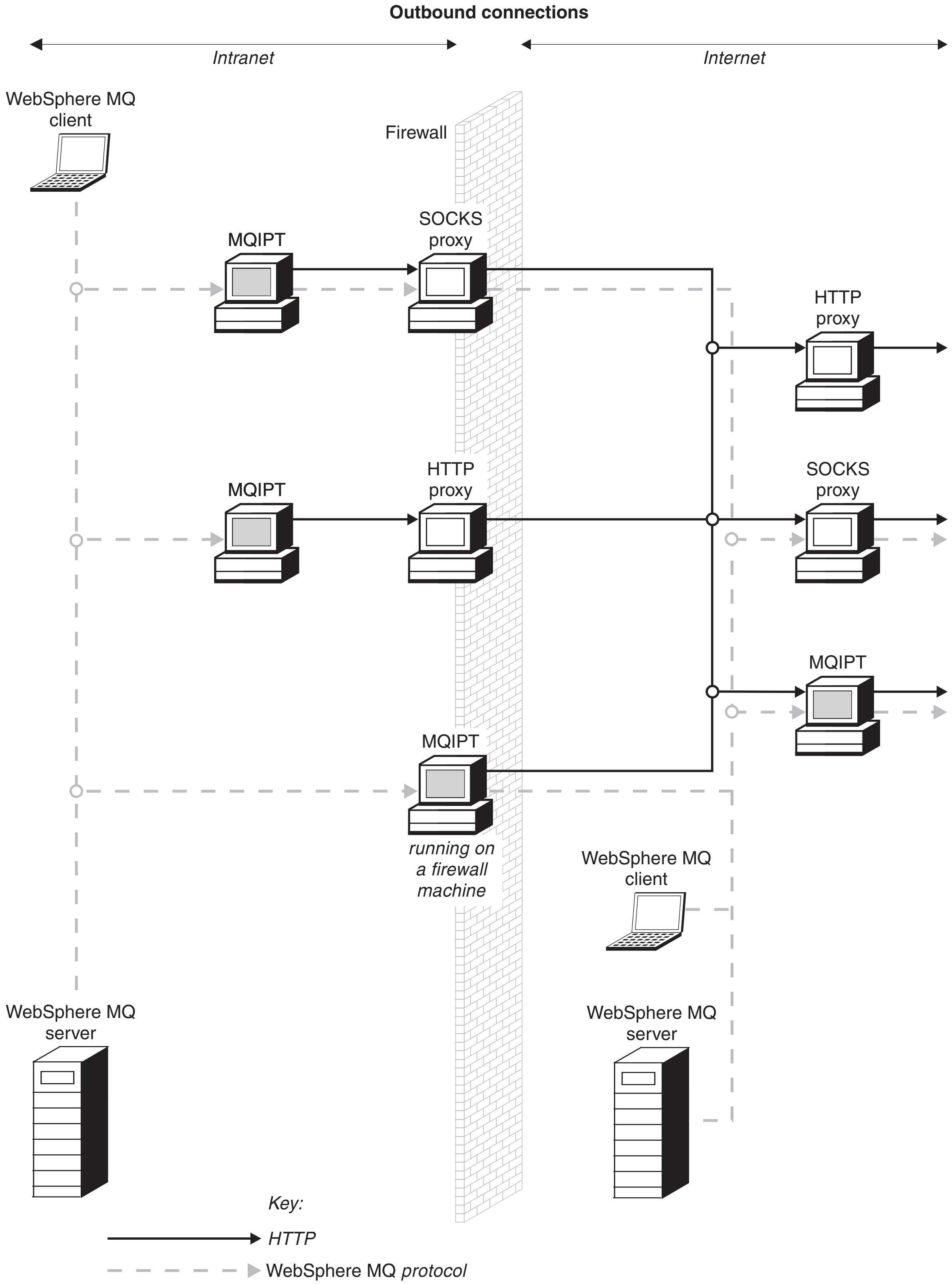


Figure 5. WebSphere MQ topology showing possible MQIPT configurations (Part 1 of 2)

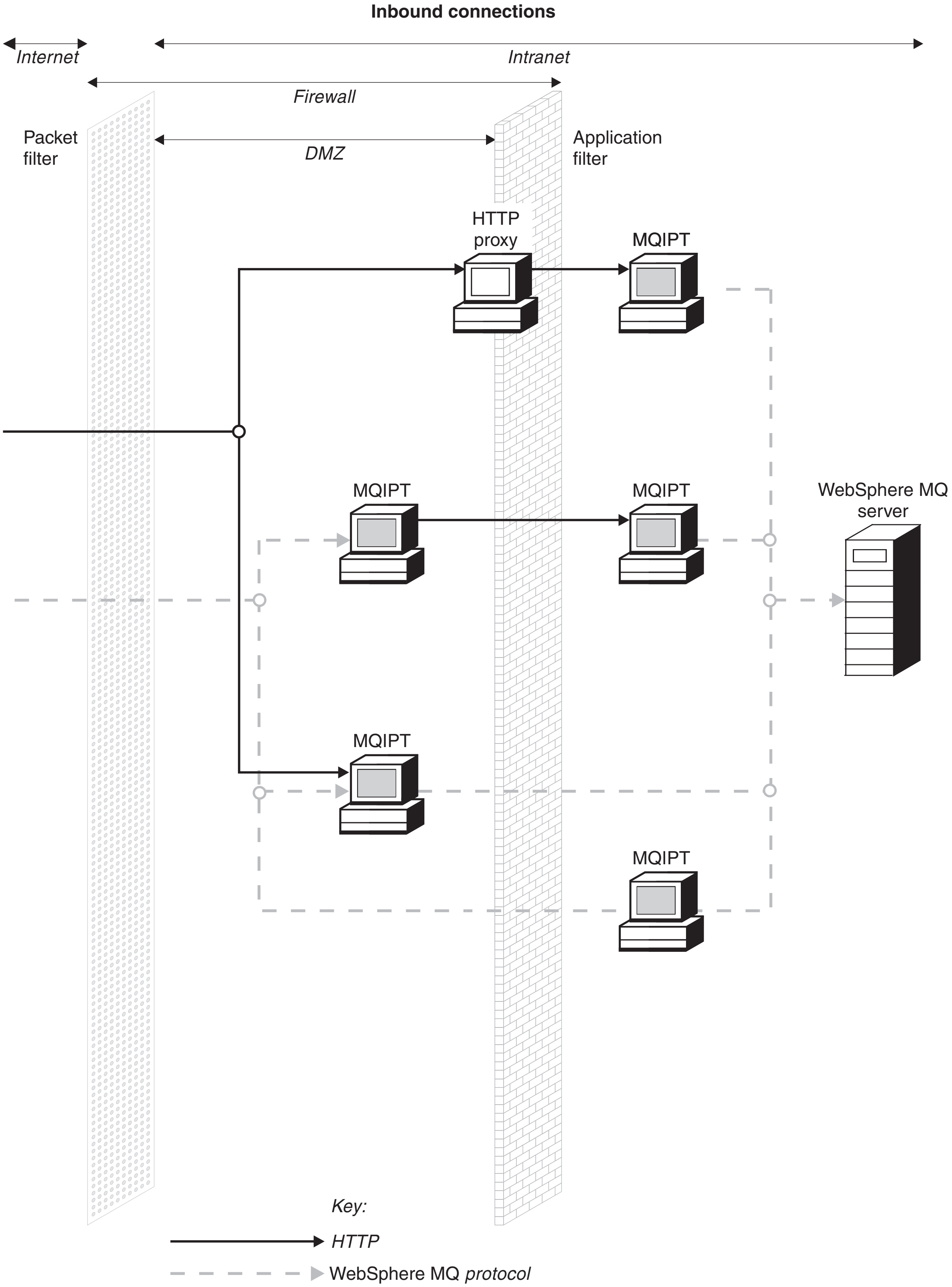


Figure 5. WebSphere MQ topology showing possible MQIPT configurations (Part 2 of 2)

Chapter 2. How internet pass-thru works

This chapter gives an overview of the way internet pass-thru works.

Overview of how internet pass-thru works

In its simplest configuration, MQIPT acts as a WebSphere MQ protocol forwarder. It listens on a TCP/IP port and accepts connection requests from WebSphere MQ channels. If a well-formed request is received, MQIPT establishes a further TCP/IP connection between itself and the destination WebSphere 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 on the original incoming connection.

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

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

MQIPT can be used to allow access to one or more destination queue managers. 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, as described in the next paragraph.

To allow access to more than one destination queue manager, MQIPT can be configured to listen on multiple TCP/IP ports. Each listening port is mapped to a destination queue manager through an MQIPT "route". The MQIPT administrator may define up to 100 such routes, which associate a listening TCP/IP port with the hostname and port of the destination queue manager. This means that the hostname (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 uses a configuration file called mqipt.conf and this file contains definitions of all routes and their associated properties. See Chapter 19, "Administering and configuring internet pass-thru", on page 71 for more information on this file.

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

Supported channel configurations

All WebSphere MQ channel types are supported, but configuration is restricted to TCP/IP connections. To a WebSphere 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 (either using HTTP tunneling, SSL, or as standard WebSphere MQ protocol packets). If MQIPT is using HTTP tunneling or SSL 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). Once 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 socksified 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 types above. 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 just like the Sender/Receiver configuration.

Chapter 3. HTTP support

MQIPT can be configured so that the data packets it forwards are encoded as HTTP requests. MQIPT supports HTTP tunneling with or without chunking.

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

When using HTTP tunneling without chunking, an HTTP reply is sent back to the first MQIPT for each HTTP request. This reply can be the response from the destination queue manager or a dummy acknowledgement. If either WebSphere MQ system has to send a chain of successive WebSphere MQ protocol packets (as happens when transferring a large message), several HTTP request/reply pairs are used to transfer the data. To achieve this, MQIPT inserts additional request or reply flows.

When using HTTP tunneling with chunking, only the first packet is wrapped in an HTTP header. Middle and last packets have chunking headers. This arrangement removes the wait for a dummy acknowledgement from the second MQIPT, and thus offers slightly better performance than that provided by HTTP tunneling without chunking.

When HTTP is being used between two MQIPTs, the TCP/IP connection on which the HTTP requests and replies are flowed is persistent and is kept open for the lifetime of the message channel. The MQIPTs do not close the TCP/IP connection between request/reply pairs.

If two MQIPTs 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 WebSphere 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) and 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.

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 reset:

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

See "HTTP proxy configuration" on page 101 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. See “HTTPS configuration” on page 116 for an example of using HTTPS.

Servlet

There is now a servlet version of MQIPT (called MQIPTServlet) that can be deployed on an Application Server as a non-distributed application. It works in a similar fashion to the normal MQIPT, but acts as though it only has one route. An incoming connection request to start a WebSphere MQ channel is handled by an instance of the MQIPTServlet and each instance maintains a persistent connection to the target queue manager. Subsequent data flows are maintained along the same channel by using the session id created during the first connection request.

A web application archive file, called MQIPTServlet.war, can be found in the web subdirectory. This war must be imported/deployed into your Application Server. If you need to specify a context name when you import this servlet, you will need to override the default UriName property to contain the new context name. See “UriName” on page 92 for more information

Configuration of the MQIPTServlet is achieved by setting properties in the web.xml file, which can be found in the WEB-INF subdirectory of the Application Server. Only a subset of the existing MQIPT properties are applicable with MQIPTServlet. The following properties can be used with MQIPTServlet:

- ClientAccess
- ConnectionLog
- MaxLogFileSize
- QMgrAccess
- Trace

Connection logs and trace files are written in a directory defined with a new property called LogDir. You are recommended to define this property before starting MQIPTServlet.

To control the amount of resources used by the MQIPTServlet, you may need to change some of the Application Server properties. Each Application Server has its own way of managing configuration data and this normally done by using a GUI, a web interface or by editing the configuration file. The properties to consider changing are the maximum number of active sessions or number of instances of the servlet within the Application Server. This will control the number of client connections and is similar to the MaxConnectionThreads property used in MQIPT.

Other properties that may need to be changed are related to timeout values, whether persistent connections are supported and how many requests are allowed on a persistent connection. As the MQIPTServlet relies on a persistent connection to the target Queue Manager, this property must be enabled. The other properties may need increasing, but will depend on their default value and the type of WebSphere MQ connection being used. WebSphere MQ client connections are normally short-lived, so it's fairly safe to use default values. Queue Manager to Queue Manager connections can run for an indeterminate length of time, in which case it is recommended some of the timeout values and the number of requests allowed on a persistent connection are increased appropriately.

There is also a session-timeout property defined in the web.xml file with a default value of 30 minutes. This property can be used to control inactivity of a client and will close a session when no activity has been detected for the specified amount of time.

There must be at least one MQIPT in the link between the client and the MQIPTServlet. The ServletClient property must be enabled in the MQIPT that connects to the MQIPTServlet, and the HTTPServer property can point either directly to the Application Server or the HTTP server that feeds the Application Server.

To test if the MQIPTServlet has started successfully, you can launch a web browser and enter a URL name similar to the following:

```
http://localhost:80/MQIPTServlet
```

a positive response will be seen in the browser.

MQIPTServlet has been tested with the IBM WebSphere Application Server 5.0 (with and without the IBM HTTP Server), Tomcat 3.3 and Tomcat 4.0. MQIPTServlet does not require Java 1.4 and will use the level of Java implemented by the Application Server.

See "Configuring the MQIPT Servlet" on page 113 for an example of how to use the servlet.

Chapter 4. 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 WMQ application to connect through MQIPT to a remote WMQ 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 WMQ clustering. See below for more information.

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

See “Configuring SOCKS proxy” on page 109 for an example of how to use SOCKS.

Clustering

WebSphere MQ clusters can be used with MQIPT by socksifying each queue manager in the cluster that spans the internet and by enabling MQIPT to act as SOCKS proxy. As there are so many different ways to configure queue managers into a cluster, the following explanation is based on the tasks described in *WebSphere MQ Queue Manager Clusters*, SC34-6061. The following diagram has been extended from that defined in the task called ″Adding a new queue manager to a cluster″. 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 socksified 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 socksified, 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 diagram below, all incoming listener port addresses are 1414 and the outgoing listener port addresses are 1415.

There are two ways to run a socksified queue manager. The first is to socksify the whole machine where the queue manager is running. The second is to socksify 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 “Configuring MQIPT Clustering support” on page 119 for an example of how to configure a cluster network.

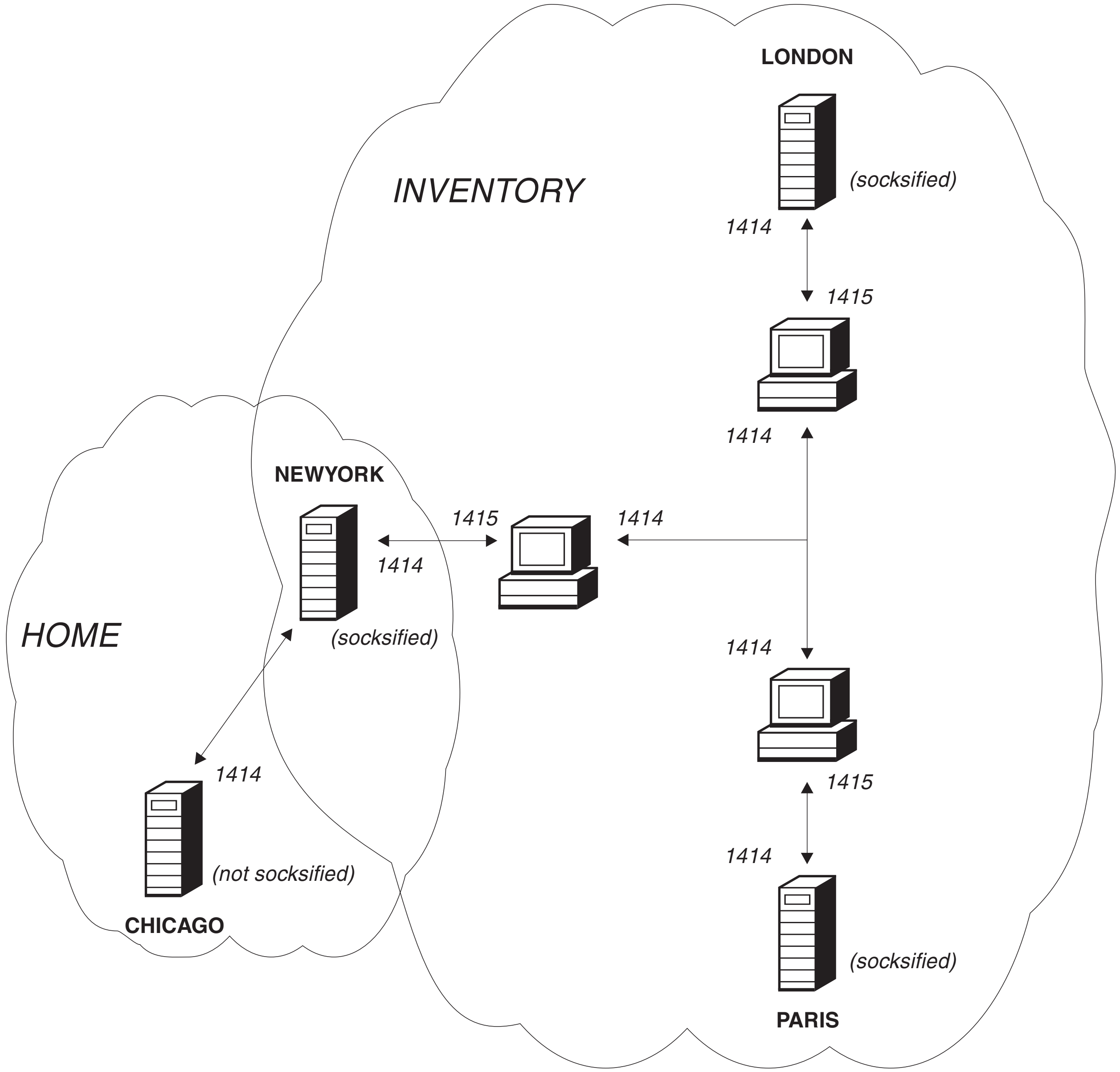


Figure 6. MQIPT Clustering support

Chapter 5. SSL overview and support

The SSL protocol provides connection security over insecure communication channels and ensures:

Communication privacy
The connection can be made private by encrypting the data to be exchanged between the client and the server, for example, only they can make sense of the data. This allows for secure transfer of private information such as credit card numbers.

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).

The SSL protocol can use different digital signature algorithms for authentication of communication parties. The cryptographic operations used in SSL, encryption for data confidentiality, and secure hashing for message integrity, rely on the sharing of secret keys between the client and the server. SSL provides various key exchange mechanisms that allow for the sharing of secret keys. SSL can make use of a variety of algorithms for encryption and hashing. Various cryptographic algorithms are supported; you specify them by using SSL cipher suites. These cipher suites are supported:

SSL_DH_anon_EXPORT_WITH_DES40_CBC_SHA
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_40_MD5
SSL_DH_anon_WITH_RC4_128_MD5
SSL_DHE_DSS_EXPORT_WITH_DES40_CBC_SHA
SSL_DHE_DSS_WITH_3DES__EDE_CBC_SHA
SSL_DHE_DSS_WITH_AES_128_CBC_SHA
SSL_DHE_DSS_WITH_AES_256_CBC_SHA
SSL_DHE_DSS_WITH_DES_CBC_SHA
SSL_DHE_RSA_EXPORT_WITH_DES40_CBC_SHA
SSL_DHE_RSA_WITH_3DES__EDE_CBC_SHA
SSL_DHE_RSA_WITH_AES_128_CBC_SHA
SSL_DHE_RSA_WITH_AES_256_CBC_SHA
SSL_DHE_RSA_WITH_DES_CBC_SHA
SSL_RSA_EXPORT_WITH_DES40_CBC_SHA
SSL_RSA_EXPORT_WITH_RC2_CBC_40_MD5#
SSL_RSA_EXPORT_WITH_RC4_40_MD5
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_NULL_MD5
SSL_RSA_WITH_NULL_SHA
SSL_RSA_WITH_RC4_128_MD5
SSL_RSA_WITH_RC4_128_SHA

SSL handshake

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

All the SSL cipher suites listed above with the exception of the anonymous cipher suites require server authentication and allow client authentication: the server can be configured to request client authentication. The communication peer authentication in SSL is based on public key cryptography and X.509v3 digital certificates. A site which should be authenticated in the SSL protocol needs 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 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 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 client inspects the server's certificate chain, starting with the site certificate, and considers the signature of the site certificate to be valid if the site certificate is in the repository of trusted site or signer certificates, or if a signer certificate in the chain can be validated based on its repository of trusted signer certificates. In the latter case the SSL 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 protocol. The SSL connection can only be established if the server really has the corresponding private key.

The client authentication follows the same procedure: if a SSL server requires client authentication the client sends to the server a certificate chain to prove its identity and the server verifies that 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 protocol. The SSL connection can only be established if the client really has the corresponding private key.

The SSL protocol itself provides very high communication security. 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 instance, if your repository of trusted site and signer certificates is compromised, you might establish a secure connection to a very insecure communication partner.

WebSphere MQ internet pass-thru and SSL

SSL V3.0 has been implemented, using 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). WebSphere MQ internet pass-thru uses the IBM Secure Socket Lite (SSLite) package.

A WebSphere MQ internet pass-thru can act as an SSL client or an SSL server depending on which end initiates the connection. The client starts a connection and the server accepts the connection request. It is possible for a WebSphere MQ internet pass-thru route to act both as a client and a server, although in this instance the use of the SSL Proxy Mode feature is recommended, for performance reasons. Each WebSphere MQ internet pass-thru route can be independently configured with its own set of SSL properties. See “Route section reference information” on page 80 for more details.

Trust settings

A key ring file contains a personal certificate including the signer certificate or chain of signer certificates. To enable authentication when a connection is being made, a certificate needs a trust setting. There are two levels of trust:

Trust as peer
Means that only this certificate may be trusted, but not any certificate signed by this certificate.

Trust as Certificate Authority (CA)
Means that any certificate signed by this certificate may be trusted.

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

The key ring file on the SSL 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 client authentication is also required, the SSLServerAskClientAuth property must be enabled on the server side and 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.

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 the PKCS#12 tokens stored in these key ring files, you must use a password of `mqiptV1.3`.

A utility called KeyMan, with which you can manage SSL certificates and key ring files, can be found in the ssl subdirectory. See “KeyMan” on page 22 for installation instructions and further information.

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

Testing SSL

Chapter 20, “Getting started with internet pass-thru”, on page 93 describes tasks that can be used to test an SSL connection.

Certificates and certificate management technologies are available from a number of vendors, including:

- RSA Security (www.rsasecurity.com)
- Entrust Technologies (www.entrust.com)
- Verisign (www.verisign.com)

SSL error messages

The following error codes can be seen in an SSLRuntimeException, if an invalid parameter value is used in one of the SSL method calls or wrong data is supplied to the SSL protocol.

Table 1. SSLRuntimeException error messages

ID	Description
1	Wrong usage of a method or that one or more input parameters are out of bounds
2	The supplied data cannot be processed
3	The signature of the supplied data cannot be verified
10	The subject name of the signer certificate does not match the issuer name of the certificate
11	The type of a certificate is not supported
12	A certificate is used before its validity period
13	A certificate is expired
14	A certificate signature cannot be verified
15	A certificate cannot be used
20	All the cipher suites proposed by the client are not supported by the server
21	All the compression methods proposed by the client are not supported by the server
22	No certificate is available
23	An algorithm or type of format is not supported
24	Rejection of obsolete information
25	A certificate is revoked
26	A set of CRLs is incomplete (some delta CRLs are missing)
27	The name to be certified already exists
28	The public key to be certified already exists
29	Some serial number or key (certificate, CRL) is wrong

Table 1. SSLRuntimeException error messages (continued)

30	Authorization failed

An SSLException is thrown if the execution of the SSL handshake protocol is terminated.

Table 2. SSLException error messages

ID	Description
3	Connect timeout defined in the SSLContext is expired and no response was received from the peer
4	Connection was aborted by peer during the SSL handshake without further error indication
10	Unexpected message was received
20	Message with a bad record MAC was received
30	Decompression failure
40	Handshake failure
41	No certificate was sent by the peer
42	Bad certificate was received
43	Unsupported certificate was received
44	Revoked certificate was received
45	Expired certificate was received
46	Unknown certificate was received
47	An illegal parameter was detected

LDAP and CRLs

WebSphere internet pass-thru 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 fashion to that in base WebSphere MQ, as the same LDAP server can possibly be used for both WebSphere MQ and MQIPT. Further information on the use of LDAP servers with WebSphere MQ can be found in the book ″WebSphere MQ Security Version 5.3″ SC34-6079-01, chapter 15. Excerpts from the book are included below for reference.

During the SSL 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 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). Subsequent references to CRLs in this chapter also apply to ARLs.

There are several proprietary LDAP directory servers in the marketplace. WebSphere internet pass-thru has been tested with the IBM Directory Server: see http://www.ibm.com/software/network/directory/server. Instructions for installing and maintaining the LDAP server will be found with the documentation provided with the installed product.

Further information on the management of CRLs and ARLs can be found in the book ″WebSphere MQ Security Version 5.3″ SC34-6079-01.

MQIPT can support up to two LDAP servers on each route. The first LDAP server is treated as the main server and the second LDAP server is seen as a backup server, and will only be 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 may be protected by a userid and password. If this is the case, the LDAP*Userid and LDAP*Password properties can be used.

When MQIPT loads a PKCS#12 token from a key ring file, any CA certificates will be checked for CRL validity. If the CA certificate has an attached CRL, it will be checked to see if it has expired and, if so, a newer CRL will be retrieved from the LDAP server. Any CRL retrieved will be 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 section reference information" on page 80).

When a query is sent to the main LDAP server, if there are no entries that match the given CA, then it will be assumed there are no CRLs for that CA. The backup server will not be used. If, however, the main LDAP server cannot be reached or does not return within a given timeframe, then the backup server will be used. Any errors from the backup server will cause the client connection to be terminated. This action can be overridden by setting the property LDAPIgnoreErrors to true.

Attention

If you enable the LDAPIgnoreErrors property a revoked certificate could be used to make an SSL connection.

The LDAP client model is based on the ″com.sun.jndi.ldap.LdapCtxFactory″ implementation. Any CRLs retrieved by MQIPT will be kept in a cache and shared by all connections on that route.

If a cached CRL has expired, the CRL will be removed from the cache and a new one will be 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 system console message is displayed (MQCPW001). The expired CRL will still be loaded into the system and any connection requests referencing that CRL will be refused. The expired CRL in the LDAP server should be replaced 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 will not be cleared until the route is restarted.

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

> **Attention**
>
> If you enable the IgnoreExpiredCRLs property a revoked certificate could be used to make an SSL connection.

The advanced encryption standard

The Advanced Encryption Standard (AES) will be a new Federal Information Processing Standard (FIPS) Publication that will specify a cryptographic algorithm for use by U.S. Government organizations to protect sensitive (unclassified) information. The National Institute of Standards and Technology (NIST) also anticipates that the AES will be widely used on a voluntary basis by organizations, institutions, and individuals outside of the U.S. Government - and outside of the United States - in some cases.

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.

Encrypting a key ring password

The password used to open a key ring 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.

Command format:

```
mqiptPW <password> <file name> <-replace>
```

where

password
: is the clear text password needed to open the given key ring file

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

replace
: is the option required to overwrite `<file name>` if it exists

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: Users who have migrated from a previous level of WebSphere Internet pass-thru will need to replace the current password files containing the cleartext password with a copy of the encrypted password file.

You must use the password `mqiptV1.3` to open either of the sample key ring files using a key management utility (for example, KeyMan).

KeyMan

A standalone utility called KeyMan is now shipped with WebSphere Internet pass-thru to allow management of SSL certificates and key ring files. A zip containing KeyMan can be found in the ssl subdirectory. To install KeyMan, unzip the file into a temporary directory and follow the instructions found in the README.txt file. KeyMan has many features, but the scope of this section is limited to creating test certificates and managing key ring files containing PKCS#12 tokens.

KeyMan is a management tool for the client side of the public key infrastructure (PKI). KeyMan manages keys, certificates, certificate revocation lists (CRLs), and the respective repositories to store and retrieve these items. The full life cycle of certificates is supported and processes involved in handling user certificates.

KeyMan manages repositories which contain collections of keys, certificates, and revocation lists. A repository is called a token. A token comprises the trust settings for a particular application (for example, WebSphere Internet pass-thru). Usually, a token contains private keys and the associated certificate chains to authenticate a user to other sites. In addition, a token holds certificates of trusted communication partners and certification authorities (CAs).

Supported types of token

KeyMan supports a number of different types of tokens. Tokens are repositories holding keys, certificates, CRLs, and trust settings. Some tokens can only store a subset of these item types.

PKCS#7 token
Contains a set of certificates and, optionally, associated CRLs. Keys cannot be stored in this type of repository. This repository does not require authentication. Certificates and CRLs are protected by a signature. However, an adversary can change the set of items as stored in a particular PKCS#7 token. This type of token is used when the expected set of items is defined by some context.

PKCS#12 token
Contains private keys, certificates, and associated CRLs. The content is protected by a user passphrase. The public items (certificates, CRLs) and the private items (keys) can be protected by algorithms with different strengths.

PKCS#11 (CryptoKi) repositories
PKCS#11 defines an interface to cryptographic tokens. These tokens can store keys and certificates. Storage of CRLs is not supported. Access to a token is protected by a personal identification number (PIN). You have to specify the token specific PKCS#11 DLL which is used by KeyMan to access the token.

KeyMan supports PKCS#11 version 2.01 and 2.10 DLLs.

PKCS#7 and PKCS#12 are soft tokens and can be retrieved from different media (for example, files, URI, and clipboard).

KeyMan has the special ability to construct PKCS#7 tokens from data with unknown format. It scans the data for X.509 certificates and CRLs and constructs a PKCS#7 token from the detected certificates and CRLs. If you have emails containing certificates or CRLs you can open the email folder in KeyMan and KeyMan will try to extract the X.509 items. Of course, the data cannot be stored back in the original format. The extracted data can be stored in a file using the PKCS#7 format.

Supported standard data formats

KeyMan supports a number of standard data formats. Following are descriptions of their meaning and usage context:

PKCS#7

This data format is a collection of certificates and CRLs. The set of certificates and CRLs as described by PKCS#7 is not protected. However, each single certificate and CRL is protected by a signature. PKCS#7 is used whenever the expected set of certificates and CRLs is defined by some context. On Windows systems the standard file suffixes for PKCS#7 files are .p7r and .p7b.

PKCS#10

PKCS#10 defines a certificate request message. It contains the public key and information about the X.500 name of the requestor. The message is signed with the corresponding private key. PKCS#10 messages can be generated in binary format and in ASCII armored format. The message must be submitted to a certificate authority (CA).

PKCS#12

PKCS#12 is used by browsers and Web servers for import and export of private keys and associated certificates. KeyMan can read and write those PKCS#12 files. While these programs understand only a very specific profile of PKCS#12 KeyMan can generate more general PKCS#12 files. KeyMan can store sets of private keys, certificates, CRLs, and corresponding trust settings in a single PKCS#12 file. PKCS#12 files are protected by a passphrase. Usually, a PKCS#12 token contains the trust policy for a particular application. In case of IBM BlueZ SSLite the keys and associated certificate chains will be used for client/server authentication. Other certificates represent trusted CAs or trusted servers depending on respective trust settings. On Windows systems the standard file suffixes for PKCS#12 files are .p12 and .pfx.

SPKAC

SignedPublicKeyAndChallenge (SPKAC) is a data format to request certificates from a CA. This particular format is generated by Netscape whenever the HTML tag <keygen> is used. It contains the signed public key and challenge. This data format can be generated by KeyMan in binary and in Base64 format.

X.509 V3 Certificates

KeyMan can read X.509 V3 certificates in binary format or wrapped in ASCII armor. These files can be opened or imported into KeyMan. It is also possible to write single certificates from a token in these two formats (**Certificate details -> Save Icon**). On Windows systems the standard file suffixes for X.509 certificate files are .crt, .cer, and .der.

X.509 V2 Certificate Revocation Lists (CRLs)
KeyMan can read X.509 V2 CRLs in binary format or wrapped in ASCII armor. A single CRL cannot be opened. KeyMan can only import CRLs into tokens which already contains the associated CA certificate. It is possible to write single CRLs in binary or ASCII armored format (**certificate details -> CRLs details -> Save Icon**). On Windows systems the standard file suffix for X.509 CRL files is .crl.

KeyMan FAQs

For general questions about cryptography and related terms refer to RSA Laboratories and their ″Frequently Asked Questions About Today's Cryptography″. The following FAQ discusses questions related to KeyMan.

Can KeyMan read PKCS#12 files generated by Netscape or Internet Explorer?
PKCS#12 files generated by Netscape browser or Internet Explorer can be read by KeyMan provided you know the password that protects their contents.

Can KeyMan create PKCS#12 files that can be read by Netscape or Internet Explorer?
The PKCS#12 standard offers a lot of freedom to chose algorithms and to arrange contents. The browsers only accept a very specific profile of all possible options. KeyMan can generate PKCS#12 files that can be read by Netscape and Internet Explorer. Since KeyMan allows you to do much more things with PKCS#12 you can create files that are not understood by these browsers. The common profile for browsers looks like this: the public/private encryption (see **Menu Options -> PKCS#12 Settings**) should be ″RC2 (40 bits)″/″DES (168 bits)″, respectively. There should be exactly one private certificate in the PKCS#12 token.

What is a private certificate?
If KeyMan detects a matching key and certificate it combines these two items into a private certificate. This means, for any private certificate you also own the corresponding private key. If you import certificates into a token KeyMan checks there is a matching private key and will automatically combine the key and the imported certificate into a private certificate. If this happens KeyMan will notify you with a dialog.

What is a CA or peer certificate?
Certificates contained in a token establish trust. They define whom you trust. What trust means and the exact evaluation of the certificates depends on the application using the token. With KeyMan you can setup two types of trust for certificates: CA and peer. If you trust a certificate as CA you implicitly trust any certificate directly or indirectly signed by this CA. If you set the trust level to ″Peer″ you trust only this particular certificate. Trust is not extended to certificates signed by a ″Peer″ certificate.

What are these certificates that are neither private, nor CA, nor peer certificates?
KeyMan tries to store for each private certificate the full chain up to the root certificate. These certificates need not be trusted and therefore will not appear among the CA or peer certificates. You can find these certificate if you select the key ring ″All Certificate Items″. The untrusted certificates do not have an icon.

What is a token?
A token is a collection of keys, certificates, and CRLs. A token is stored on some media (for example, a file, a URL, piece of hardware). There are

different types of tokens with different capabilities: software tokens, hardware tokens, unprotected tokens, and tokens protected by passwords or PINs.

What is a key ring?

A token consists of a set of key rings. A particular key ring identifies a specific set of items (for example, certificates of the same trust level, or certificates for which you own the private key, or keys without matching certificates).

Chapter 6. Quality of service

Quality of Service (QoS)

The IBM WebSphere Edge Server provides a bandwidth management solution through the Transactional Quality of Service plug-in on the Linux platform. Transactional Quality of Service (TQoS) refers to the overall service, in terms of elements such as throughput and delay, that is provided to network users. Attributes can be set to assure a quality of service associated with any outgoing data being sent along a connection. This allows the policy administrator to define rules that identify traffic related to specific servers and policy actions with unique differentiated service controls for this traffic. For example, an installation can define a policy that specifies preferential treatment of outgoing traffic related to the server traffic in support of a sale of a certain amount of goods as opposed to server traffic in support of a client browsing. Additionally, TQoS also allows administrators to collect performance data of the corresponding policy to monitor if the policy delivers the service level objectives (important measurements such as connection throughput, delays, loss ratio and so on) for which they are intended. MQIPT only requires the Policy Agent (pagent) to be installed and running to implement a Quality of Service (QoS).

TQoS policies are defined in a policy configuration file (pagent.conf) or by using an LDAP server. The TQoS pagent can either access the policy configuration file, or go to an LDAP server, or both to retrieve TQoS policy entries. The *IBM Edge Server Administration Guide* gives more information on pagent, it can be found at the following URL:

```
http://www.ibm.com/software/webservers/edgeserver/library.html
```

From this site you can either view the HTML online or download the PDF version, in either format you can perform a search for TQoS.

The TQoS code, together with installation and administration instructions, can be downloaded from the same location as MQIPT. See the WebSphere MQ family SupportPacs site at http://www.ibm.com/webspheremq/supportpacs and click Category 3 – Product Extensions.

MQIPT is shipped with a dummy library called libmqiptqos.so, found in the MQIPT lib subdirectory. This enables MQIPT to be run on the Linux platform without having to install the TQoS pagent. After installing TQoS, you may need to replace this dummy library with that used by TQoS. A script called mqiptQoS can be found in the MQIPT bin subdirectory to help with this task. Use the following command to rename the dummy library and define a soft link to the real TQoS runtime library:

```
mqiptQoS -install
```

Using mqiptQoS -remove will reverse the above actions.

MQIPT only requires the pagent to be installed and running to implement a Quality of Service (QoS). Using MQIPT, an application priority can be set on a route for data flowing in each direction and this will, therefore, affect all channels using that route. The priority is defined using the MQIPT properties QosToCaller and QosToDest (see "Route section reference information" on page 80 for more information) and the values used here must match an ApplicationPriority policy definition in the `pagent.conf` control file. If the pagent does not find a matching policy, then the data will not be assigned any priority. Any changes to a policy will not be reflected to MQIPT until the pagent has been restarted. See "Configuring Quality of Service (QoS)" on page 105 for more information on policy definitions.

Chapter 7. Network dispatcher

Network Dispatcher support

MQIPT can be used with the 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* (GC31-8496) for installing custom advisors. Figure 7 shows an example of the use of the Network Dispatcher for monitoring port address 1414 for MQIPT. Note that each MQIPT must have the same configuration file.

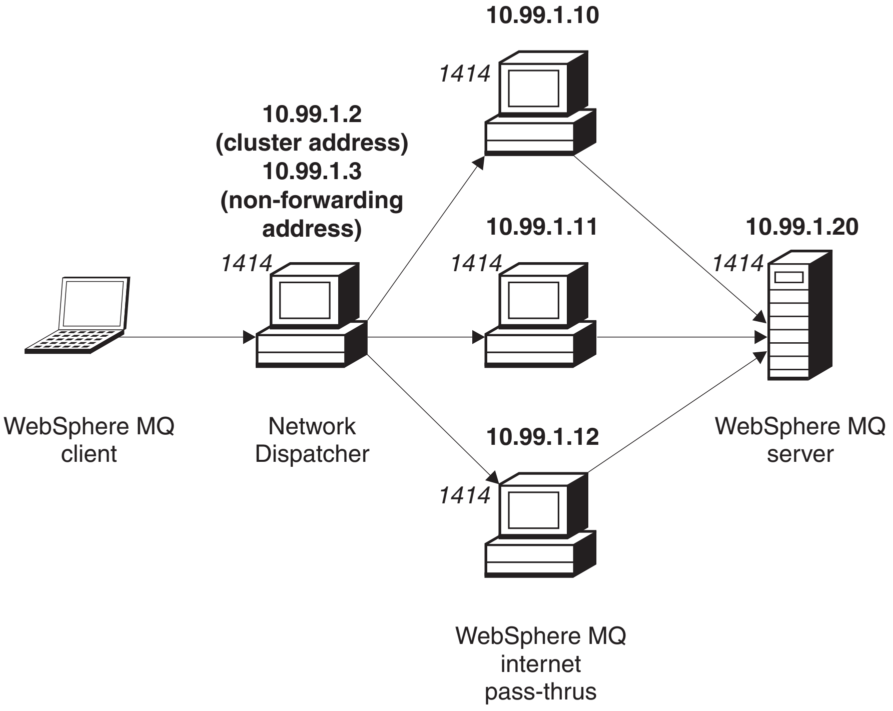


Figure 7. Using the Network Dispatcher with MQIPT

Follow the instructions in Chapter 5 of the *Network Dispatcher User's Guide* for configuring the dispatcher component to define port 1414 and the load-balanced server machines. 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 this 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 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:\winnt\system32 directory (on Windows NT). See the mqipt1414Sample.ssl file for more information.

Chapter 8. Java security manager and security exits

Java Security Manager

Support of the Java Security Manager was originally implemented for use with the SSL proxy mode feature to manage the control of socket connections, but it can also be used with any of the other MQIPT features 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 section reference information" on page 80 for more information.

The Java Security Manager uses two default policy files. A global system policy file named `$JREHOME/lib/security/java.policy` (where `$JREHOME` is the directory that contains your Java Runtime environment) is used by all instances of a virtual machine on a host. A second, user-specific policy file called `.java.policy` may exist in the user's home directory. An additional MQIPT policy file can also be used, see "Global section reference information" on page 80 for more information. To use an additional policy file, make sure the `policy.allowSystemProperty` property has been set to true in the global system policy file (`java.security`).

The syntax of the policy file is quite complex and although it can be changed using a text editor, it is recommended you use the policytool utility provided with Java for making any changes. The policytool utility can be found in the `$JREHOME/bin` directory and is fully documented within the Java documentation.

A sample policy file (`mqiptSample.policy`) has been provided with MQIPT to show which permissions need to be set for running MQIPT. It is only the `java.net.SocketPermission` entries that need adding/changing/deleting to match your own specific requirements to control who can connect to MQIPT and who MQIPT can connect to. This sample file assumes MQIPT has been installed in the default home directory, for example, `c:\Program Files\IBM\Websphere MQ internet pass-thru\`. If you have installed MQIPT in another location you will need to reflect this in the `codeBase` and `java.io.FilePermission` definitions.

Permissions are usually defined with three attributes and 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 hostname 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`
- `*`

- `123.456.789` (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.

Security exit

Attention

MQIPT runs in a single JVM so a user-defined security exit could jeopardize MQIPT's normal operation by:

- affecting system resources
- generating bottlenecks
- degrading performance

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

The purpose of a security exit is to control access to a target destination, as defined by the Destination route property. The security exit will be called at the point when a connection request has been received from a client and before MQIPT makes the connection to the target destination. Based on the initial connection properties, the security exit can decide if the connection will be allowed to complete.

When a route is started, the security exit will be called to initialize and 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. Finally, 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.

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

Three sample security exits have been provided to show how a security exit can be implemented. The first sample, called SampleSecurityExit, shows how to control access to a WebSphere MQ Queue Manager, based on the name of the WMQ channel. It will only allow a connection with a channel name starting with the string ″MQIPT.″ See “Security exit” on page 136 for more information.

The second sample, called SampleRoutingExit, allows dynamic routing of client connection requests to a pool of defined WebSphere MQ servers, each server hosting a QM of the same name and same attributes. The sample includes a configuration file that contains a list of server names. See “Routing security exit” on page 138 for more information.

The third sample, called SampleOneRouteExit, allows dynamic routing to a WMQ QM that is derived from the WMQ channel name used in the connection request. The sample includes a configuration file that contains a map of QM names to server names. See “Dynamic one route exit” on page 141 for more information.

The com.ibm.mq.ipt.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. See below for a complete list of supported methods.

Methods

init

```
public void init () throws IPTException
```

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. Any exception thrown in this method will prevent the route from starting.

refresh

```
public void refresh () throws IPTException
```

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). Any exception thrown in this method will cause the route to be disabled.

close

```
public void close ()
```

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.

validate

```
public SecurityExitResponse validate ()
```

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 data and therefore the data usually obtained from the initial data flow will be unreadable. The queue manager name will not be available for WMQ client connections, as this information is not available until after the connection to the target Queue Manager has been established.

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 (QM). 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 WMQ channel name used in the connection request

public String getQMName()
retrieves the WMQ 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.SecurityExitResponse class

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

Creating a default SecurityExitResponse object will reject the connection request.

Supported constructors:

public SecurityExitResponse (String dest, int destPort, int rc, String msg) throws IPTException

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) throws IPTException

public SecurityExitResponse (int rc, String msg) throws IPTException

public SecurityExitResponse (int rc) throws IPTException

Supported methods for setting property values:

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) throws IPTException
sets the reason code for the connection request - throw an IPTException for an unknown value

Valid reason codes:

- `SecurityExitResponse.OK = 0`
- `SecurityExitResponse.NOT_AUTHORIZED = 1`
- `SecurityExitResponse.NOT_READY = 2`

Tracing

To help diagnose any problems in a user-defined security exit you can enable a trace facility, similar to that used by MQIPT. Setting the route Trace property to a value of 1-5 will create a trace file in the errors subdirectory. The name of the trace file is the same as that of the security exit.

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.

Initialization of the tracing functions is performed by MQIPT when the security exit is started; all you must do is choose what information you would like to trace. There are many tracing examples in the sample user exits.

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

```
<a_method>
{
  SecurityExit.rastlRoute.entry(RASITraceEvent.TYPE_ENTRY_EXIT,
                                this,
                                "method_name");
⋮
  <code>
⋮
  SecurityExit.rastlRoute.trace(RASITraceEvent.TYPE_MISC_DATA,
                                this,
                                "data");
⋮
  <code>
⋮
  SecurityExit.rastlRoute.exit(RASITraceEvent.TYPE_ENTRY_EXIT,
                               this,
                               "method_name");
}
```

Chapter 9. Port address control

Port address control

When using MQIPT, it is possible to restrict the range of local port addresses it uses when making an outgoing connection by setting the OutgoingPort property on the route. The range of local port addresses is calculated by using the MaxConnectionThreads value. For example, if OutgoingPort is set to 1600 and MaxConnectionThreads is set to 20, then the range of local port addresses, for that route, would be 1600-1619. It is the responsibility of the MQIPT administrator to make sure there are no conflicts of port addresses across routes. If OutgoingPort is not defined, a default value of 0 means a system allocated port address will be used for each connection.

See the example, “Allocating port addresses” on page 125, 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.

Chapter 10. Other security considerations

Other security considerations

If you choose not to use SSL, MQIPT allows channel security flows, so that WebSphere MQ channel exits can be used to provide security over the entire channel from end to end.

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 machine.
- 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 WebSphere MQ protocol. This helps prevent MQIPTs being used for security attacks outside of the WebSphere MQ protocol. If MQIPT is acting as an SSL proxy, when all WebSphere MQ data and protocols have been encrypted MQIPT can only guarantee the initial SSL handshake. In this situation you are recommended to use the Java Security Manager, see "Java Security Manager" on page 31.
- It allows channel exits to run their own end-to-end security protocols.
- MQIPT allows you to 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's configuration file, `mqipt.conf`, because this file controls access to the internal hosts, and you must prevent unauthorized access to the command port (if it is enabled) because such access allows an external person to shut down MQIPT.

Chapter 11. Miscellaneous features

Normal termination and failure conditions

When MQIPT detects closure (either normal or abnormal) of a WebSphere MQ channel, it propagates the channel closure. If the administrator closes down a route through the 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 two WebSphere MQ systems at either end of the channel observe these abnormal termination conditions either as network failures, or as termination of the channel by their partner. The channels in question are then able to restart and recover (if the failure happens during a protocol in-doubt period) just as they would do if there were no MQIPTs being used.

Safety of messages

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

See *MQSeries Intercommunication*, SC33-1872 for more information on WebSphere MQ messages and channels.

Connection logs

MQIPT provides a connection log facility which contains lists of all successful and unsuccessful connection attempts. It is controlled using the `ConnectionLog` and `MaxLogFileSize` properties. See “Global section reference information” on page 80 for more information.

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

```
mqiptYYYYMMDDHHmmSS.log
```

where

- `YYYY` is the year
- `MM` is the month
- `DD` is the day
- `HH` is the hour
- `mm` is the minute
- `SS` is the second

For audit purposes, these log files are never erased. It is the responsibility of the MQIPT administrator to manage these files and delete them when they are no longer required.

Chapter 12. Upgrading from the previous version

To upgrade MQIPT from Version 1.2 to Version 1.3, follow these steps:

1. Take a copy of the configuration files `mqipt.conf` and `client.conf`. `mqipt.conf` can be found in the MQIPT home directory and `client.conf` in the bin subdirectory.
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:
   ```
   mqiptService -remove
   ```
4. Run the uninstallation program for MQIPT.
5. After you have installed MQIPT V1.3, copy the saved configuration files back to their original locations.
6. You are advised to use the MQIPT Administration GUI to manage changes to MQIPT. The configuration file from V1.2 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.

It is also possible to upgrade the core of MQIPT without uninstalling it first. All the classes needed to run MQIPT are stored in the MQipt.jar file; you can install the latest version of MQIPT on another machine and copy the MQipt.jar file from that installation to your live system. The same is true of the classes needed to run the Administration GUI. These are contained in the guiadmin.jar file.

New configuration options

The following properties are new in Version 1.3:

- IgnoreExpiredCRLs
- LDAP
- LDAPCacheTimeout
- LDAPIgnoreErrors
- LDAPSaveCRL
- LDAPServer1
- LDAPServer1Password
- LDAPServer1Port
- LDAPServer1Timeout
- LDAPServer1Userid
- LDAPServer2
- LDAPServer2Password
- LDAPServer2Port
- LDAPServer2Timeout

- LDAPServer2Userid
- RouteRestart
- SecurityExit
- SecurityExitName
- SecurityExitPath
- SecurityExitTimeout
- SSLClientSiteDN_C
- SSLClientSiteDN_CN
- SSLClientSiteDN_L
- SSLClientSiteDN_O
- SSLClientSiteDN_OU
- SSLClientSiteDN_ST
- SSLClientSiteLabel
- SSLServerSiteDN_C
- SSLServerSiteDN_CN
- SSLServerSiteDN_L
- SSLServerSiteDN_O
- SSLServerSiteDN_OU
- SSLServerSiteDN_ST
- SSLServerSiteLabel

For reference information about all the properties, see “Configuration reference information” on page 76.

Chapter 13. Installing internet pass-thru on Windows

This chapter describes how you install MQIPT on a Windows NT, Windows 2000, or Windows XP system:

- "Downloading and installing the files"
- "Setting up internet pass-thru" on page 48
- "Starting internet pass-thru from the command line" on page 48
- "Starting the Administration Client from the command line" on page 49
- "Using a Windows service control program" on page 49
- "Uninstalling internet pass-thru as a Windows service" on page 50
- "Uninstalling internet pass-thru" on page 50

Downloading and installing the files

MQIPT (MS81, a category 3 SupportPac™) can be downloaded from the WebSphere MQ SupportPac Web page, at:

`http://www.ibm.com/webspheremq/supportpacs`

Follow the instructions for downloading.

Open a command prompt and unpack `ms81_nt.zip` into a temporary directory. Run the `setup.exe` and follow the online instructions.

MQIPT must be installed by a user with Administrator authority.

MQIPT contains the files shown in the following table and the files for the Administration Client GUI, shipped as a separately installable feature, shown in the next table.

File	Purpose
Readme.txt	Latest information not included in the publications
mqiptSample.conf	Sample configuration file
ssl\sslSample.pfx	Test key ring file
ssl\sslSample.pwd	Password file for test key ring file
ssl\sslCAdefault.pfx	Sample Certificate Authority (CA) key ring file
ssl\sslCAdefault.pwd	Password file for sample CA key ring file
ssl\KeyMan.zip	KeyMan utility
exits\ SampleOneRouteExit.java	Sample security exit
exits\ SampleOneRouteExit.conf	Configuration file for SampleOneRouteExit
exits\SampleRoutingExit.java	Sample security exit
exits\SampleRoutingExit.conf	Configuration file for SampleRoutingExit
exits\SampleSecurityExit.java	Sample security exit
lib\MQipt.jar	Contains runtime, class, and property files
lib\ADV_mqipt_normal. class	Network Dispatcher advisor for "normal" mode

File	Purpose
lib\ADV_mqipt_replace. class	Network Dispatcher advisor for “replace” mode
lib\mqipt1414Sample.ssl	Sample trigger file for Network Dispatcher advisor
bin\mqipt.bat	Shortcut for running MQIPT from the command line
bin\mqiptAdmin.bat	Shortcut for stopping MQIPT and refreshing file information
bin\mqiptPW.bat	Encrypting password used to open key ring files
bin\mqiptservice.exe	For adding or removing MQIPT to or from the Windows Service Control Manager
bin\mqiptVersion.bat	Displays the version number of MQIPT
web\MQIPTServlet.war	Web archive file for servlet version.
doc\<lang>\html\<filename>.zip	Master file for the *internet pass-thru* manual in HTML format. See “Bibliography” on page 169 for more information about softcopy documentation.

The files associated with the Administration Client GUI feature are:

File	Purpose
lib\guiadmin.jar	Contains runtime, class and property files
bin\mqiptGui.bat	Shortcut for running the Administration Client from the command line
bin\customSample.properties	Sample file for customizing the appearance and, therefore, accessibility of the Administration Client

The installer updates the system CLASSPATH environment variable with the location of the `MQipt.jar` and `guiadmin.jar` files.

Setting up internet pass-thru

Before starting MQIPT for the first time, copy the sample configuration file `mqiptSample.conf`, to `mqipt.conf`. See Chapter 19, “Administering and configuring internet pass-thru”, on page 71 for further information.

Starting internet pass-thru from the command line

Open a command prompt and change directory to the `bin` directory and run `mqipt`. For example:

```
c:
cd \mqipt\bin
mqipt ..
```

You can also start MQIPT from the Windows Start -> Programs menu.

Running the `mqipt` script without any options uses a default location of “.” for the configuration file (`mqipt.conf`). To specify a different location:

```
mqipt <directory name>
```

Messages will appear on the console showing the status of MQIPT. If an error occurs, see “Problem determination” on page 145. The following messages are an example of MQIPT successfully starting:

```
5639-L92 (C) Copyright IBM Corp. 2000, 2003 All Rights Reserved
MQCPI001 WebSphere MQ internet pass-thru Version 1.3.0 starting
MQCPI004 Reading configuration information from c:\mqipt\mqipt.conf
MQCPI008 Listening for control commands on port 1881
MQCPI011 The path c:\mqipt\logs will be used to store the log files
MQCPI006 Route 1418 has started and will forward messages to :
MQCPI034 ....mqserver.company4.com(1414)
MQCPI035 ....using MQ protocols
MQCPI078 Route 1418 ready for connection requests
MQCPI006 Route 1415 has started and will forward messages to :
MQCPI034 ....mqipt.company2.com(1415)
MQCPI035 ....using MQ protocols
MQCPI036 ....SSL Client side enabled with properties :
MQCPI031 ......cipher suites <null>
MQCPI032 ......keyring file c:\mqipt\KeyMan.pfx
MQCPI038 ......distinguished name(s) CN=*Doe O=IBM OU=* L=* ST=* C=*
MQCPI078 Route 1415 ready for connection requests
```

The following subdirectories of the mqipt home directory are created automatically when MQIPT is invoked for the first time:

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

Starting the Administration Client from the command line

Open a command prompt and change directory to the bin directory and run mqiptGui. For example:

```
c:
cd \mqipt\bin
mqiptGui
```

To allow the Administration Client to connect outwards through a firewall to an MQIPT using a SOCKS proxy, specify the host name or address and port number:

```
mqiptGui <socksHostName <socksPort>>
```

The default socksPort is 1080.

The status of the Administration Client is shown by messages appearing in the Administration Client's main window.

Using a Windows service control program

A separate service control program, mqiptservice.exe, is provided to allow MQIPT to be managed and started as a Windows service.

mqiptservice.exe takes the following command line arguments:

mqiptservice -install *path*
Installs and registers the service, so that it appears on the Windows services panel as a manual service. Go to the services panel and change the setting to “automatic” to make MQIPT start automatically when the system starts. You have to 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. Put quotes around the path name if the name contains blanks.

mqiptservice -remove
Removes the service, making it disappear from the services panel.

mqiptservice ?
Displays US English help messages listing the valid arguments.

Specifying both install and remove on the same command causes an error.

Windows internally invokes the `mqiptservice` program with no arguments. If you call it 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 up. When it is stopped, all routes are subjected to immediate shutdown.

Note: The system PATH environment variable must contain the location of the JNI runtime libraries. The `jvm.dll` file can be found in the `client` subdirectory of the JDK.

Uninstalling internet pass-thru as a Windows service

You uninstall MQIPT as a service by first stopping it from the Windows services panel. Then open a command prompt, go to MQIPT's `bin` subdirectory, and type:

```
mqiptservice -remove
```

Uninstalling internet pass-thru

Before uninstalling MQIPT from your system, remove it as a Windows Service, as described above. Then run the uninstall process from the Windows Start menu.

Chapter 14. Installing internet pass-thru on Sun Solaris

This chapter describes how you install MQIPT on a Sun Solaris system:

- "Downloading and installing the files"
- "Setting up internet pass-thru" on page 52
- "Starting internet pass-thru from the command line" on page 52
- "Starting internet pass-thru automatically" on page 53
- "Starting the Administration Client from the command line" on page 53
- "Uninstalling internet pass-thru" on page 53

Downloading and installing the files

MQIPT can be downloaded from the WebSphere MQ SupportPac Web page, at:

```
http://www.ibm.com/webspheremq/supportpacs
```

Follow the instructions for downloading.

Log in as `root`, uncompress and unpack `ms81_sol.tar.Z` into a temporary directory. Run the `pkgadd` command, as in this example:

```
login root
cd /tmp
uncompress -fv ms81_sol.tar.Z
tar xvf ms81_sol.tar
pkgadd -d . mqipt
```

The example assumes that `ms81_sol.tar.Z` is in the /tmp directory.

MQIPT contains the files shown in the following table, including the files for the Administration Client GUI.

File	Purpose
Readme.txt	Latest information not included in the publications
mqiptSample.conf	Sample configuration file
ssl/sslSample.pfx	Test key ring file
ssl/sslSample.pwd	Password file for test key ring file
ssl/sslCAdefault.pfx	Sample Certificate Authority (CA) key ring file
ssl/sslCAdefault.pwd	Password file for sample CA key ring file
ssl/KeyMan.zip	KeyMan utility
exits/ SampleOneRouteExit.java	Sample security exit
exits/ SampleOneRouteExit.conf	Configuration file for SampleOneRouteExit
exits/SampleRoutingExit.java	Sample security exit
exits/SampleRoutingExit.conf	Configuration file for SampleRoutingExit
exits/SampleSecurityExit.java	Sample security exit
lib/MQipt.jar	Contains runtime, class, and property files
lib/ADV_mqipt_normal. class	Network Dispatcher advisor for "normal" mode

File	Purpose
lib/ADV_mqipt_replace. class	Network Dispatcher advisor for “replace” mode
lib/mqipt1414Sample.ssl	Sample trigger file for Network Dispatcher advisor
bin/mqipt	Shortcut for running MQIPT from the command line
bin/mqiptAdmin	Shortcut for stopping MQIPT and refreshing file information
bin/mqiptPW	Encrypting password used to open key ring files
bin/mqiptVersion	Display the version number of MQIPT
bin/mqiptService	For installing MQIPT so that it starts automatically at system startup.
bin/mqiptEnv	Defines the location of the mqipt.jar file and is used only by the other scripts.
web/MQIPTServlet.war	Web archive file for servlet version.
doc/<lang>/html/ <filename>.zip	Master file for the *internet pass-thru* manual in HTML format. See “Bibliography” on page 169 for more information about softcopy documentation.
lib/guiadmin.jar	Contains runtime, class and property files for the Administration Client GUI
bin/mqiptGui	Shortcut for running the Administration Client GUI from the command line
bin/customSample. properties	Sample file for customizing the appearance and, therefore, accessibility of the Administration Client

Setting up internet pass-thru

Before starting MQIPT for the first time, copy the sample configuration file, mqiptSample.conf, to mqipt.conf. See Chapter 19, “Administering and configuring internet pass-thru”, on page 71 for further information.

Starting internet pass-thru from the command line

Log in as root and change directory to the bin directory. For example:

```
cd /opt/mqipt/bin
mqipt ..
```

Running the mqipt script without any options uses a default location of “.” for the configuration file (mqipt.conf). To specify a different location:

```
mqipt <directory name>
```

Messages will appear on the console showing the status of MQIPT. If an error occurs, see “Problem determination” on page 145. The following messages are an example of MQIPT successfully starting:

```
5639-L92 (C) Copyright IBM Corp. 2000, 2003 All Rights Reserved
MQCPI001 WebSphere MQ internet pass-thru Version 1.3.0 starting
MQCPI004 Reading configuration information from /opt/mqipt/mqipt.conf
MQCPI008 Listening for control commands on port 1881
MQCPI011 The path /opt/mqipt/logs will be used to store the log files
MQCPI006 Route 1418 has started and will forward messages to :
MQCPI034 ....mqserver.company4.com(1414)
MQCPI035 ....using MQ protocols
MQCPI078 Route 1418 ready for connection requests
MQCPI006 Route 1415 has started and will forward messages to :
```

```
MQCPI034 ....mqipt.company2.com(1415)
MQCPI035 ....using MQ protocols
MQCPI036 ....SSL Client side enabled with properties :
MQCPI031 ......cipher suites <null>
MQCPI032 ......keyring file /opt/mqipt/KeyMan.pfx
MQCPI038 ......distinguished name(s) CN=*Doe O=IBM OU=* L=* ST=* C=*
MQCPI078 Route 1415 ready for connection requests
```

The following subdirectories of the `mqipt` home directory are created automatically when MQIPT is invoked for the first time:

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

Starting internet pass-thru automatically

To start MQIPT automatically when the system is started, run the `mqiptService` script. For example:

```
cd /opt/mqipt/bin
mqiptService -install
```

To prevent MQIPT from starting automatically:

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

Starting the Administration Client from the command line

Open a command prompt and change directory to the `bin` directory and run `mqiptGui`. For example:

```
cd /opt/mqipt/bin
mqiptGui
```

To allow the Administration Client to connect outwards through a firewall to an MQIPT, specify the host name or address and port number:

```
mqiptGui <socksHostName <socksPort>>
```

The default `socksPort` is 1080.

The status of the Administration Client is shown by messages appearing in the Administration Client’s main window.

Uninstalling internet pass-thru

Before uninstalling MQIPT from your system, prevent it from starting automatically, as described in “Starting internet pass-thru automatically”. Log in as `root` and run the `pkgrm` command:

```
pkgrm mqipt
```

Chapter 15. Installing internet pass-thru on AIX

This chapter describes how you install MQIPT on an AIX system:

- "Downloading and installing the files"
- "Setting up internet pass-thru" on page 56
- "Starting internet pass-thru from the command line" on page 56
- "Starting internet pass-thru automatically" on page 57
- "Starting the Administration Client from the command line" on page 57
- "Uninstalling internet pass-thru" on page 57

Downloading and installing the files

MQIPT can be downloaded from the WebSphere MQ SupportPac Web page, at:

```
http://www.ibm.com/webspheremq/supportpacs
```

Follow the instructions for downloading.

Log in as `root`, uncompress and unpack `ms81_aix.tar.Z` into a temporary directory. Run the `installp` command, as in this example:

```
cd /tmp
uncompress -fv ms81_aix.tar.Z
tar xvf ms81_aix.tar
installp -d . -a mqipt-RT
```

The example assumes that `ms81_aix.tar.Z` is in the /tmp directory.

MQIPT contains the files shown in the following table, including the files for the Administration Client GUI.

File	**Purpose**
Readme.txt	Latest information not included in the publications
mqiptSample.conf	Sample configuration file
ssl/sslSample.pfx	Test key ring file
ssl/sslSample.pwd	Password file for test key ring file
ssl/sslCAdefault.pfx	Sample Certificate Authority (CA) key ring file
ssl/sslCAdefault.pwd	Password file for sample CA key ring file
ssl/KeyMan.zip	KeyMan utility
exits/ SampleOneRouteExit.java	Sample security exit
exits/ SampleOneRouteExit.conf	Configuration file for SampleOneRouteExit
exits/SampleRoutingExit.java	Sample security exit
exits/SampleRoutingExit.conf	Configuration file for SampleRoutingExit
exits/SampleSecurityExit.java	Sample security exit
lib/MQipt.jar	Contains runtime, class, and property files
lib/ADV_mqipt_normal. class	Network Dispatcher advisor for "normal" mode

File	Purpose
lib/ADV_mqipt_replace. class	Network Dispatcher advisor for “replace” mode
lib/mqipt1414Sample.ssl	Sample trigger file for Network Dispatcher advisor
bin/mqipt	Shortcut for running MQIPT from the command line
bin/mqiptAdmin	Shortcut for stopping MQIPT and refreshing file information
bin/mqiptPW	Encrypting password used to open key ring files
bin/mqiptVersion	Display the version number of MQIPT
bin/mqiptService	For installing MQIPT so that it starts automatically at system startup.
bin/mqiptEnv	Defines the location of the `mqipt.jar` file and is used only by the other scripts.
web/MQIPTServlet.war	Web archive file for servlet version
doc/<lang>/html/ <filename>.zip	Master file for the *internet pass-thru* manual in HTML format. See “Bibliography” on page 169 for more information about softcopy documentation.
lib/guiadmin.jar	Contains runtime, class and property files for the Administration Client GUI
bin/mqiptGui	Shortcut for running the Administration Client from the command line
bin/customSample. properties	Sample file for customizing the appearance and, therefore, accessibility of the Administration Client

Setting up internet pass-thru

Before starting MQIPT for the first time, copy the sample configuration file, `mqiptSample.conf`, to `mqipt.conf`. See Chapter 19, “Administering and configuring internet pass-thru”, on page 71 for further information.

Starting internet pass-thru from the command line

Log in as `root` and change directory to the `bin` directory. For example:

```
cd /usr/opt/mqipt/bin
mqipt ..
```

Running the `mqipt` script without any options uses a default location of “.” for the configuration file (`mqipt.conf`). To specify a different location:

```
mqipt <directory name>
```

Messages will appear on the console showing the status of MQIPT. If an error occurs, see “Problem determination” on page 145. The following messages are an example of MQIPT successfully starting:

```
5639-L92 (C) Copyright IBM Corp. 2000, 2003 All Rights Reserved
MQCPI001 WebSphere MQ internet pass-thru Version 1.3.0 starting
MQCPI004 Reading configuration information from /usr/opt/mqipt/mqipt.conf
MQCPI008 Listening for control commands on port 1881
MQCPI011 The path /usr/opt/mqipt/logs will be used to store the log files
MQCPI006 Route 1418 has started and will forward messages to :
MQCPI034 ....mqserver.company4.com(1414)
MQCPI035 ....using MQ protocols
MQCPI078 Route 1418 ready for connection requests
MQCPI006 Route 1415 has started and will forward messages to :
```

```
MQCPI034 ....mqipt.company2.com(1415)
MQCPI035 ....using MQ protocols
MQCPI036 ....SSL Client side enabled with properties :
MQCPI031 ......cipher suites <null>
MQCPI032 ......keyring file /usr/opt/mqipt/KeyMan.pfx
MQCPI038 ......distinguished name(s) CN=*Doe O=IBM OU=* L=* ST=* C=*
MQCPI078 Route 1415 ready for connection requests
```

The following subdirectories of the mqipt home directory are created automatically when MQIPT is invoked for the first time:

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

Starting internet pass-thru automatically

To start MQIPT automatically when the system is started, run the mqiptService script to add an entry in the inittab. For example:

```
cd /usr/opt/mqipt/bin
../mqiptService -install
```

To prevent MQIPT from starting automatically and remove its entry from inittab:

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

Starting the Administration Client from the command line

Open a command prompt and change directory to the bin directory and run mqiptGui. For example:

```
cd /usr/opt/mqipt/bin
../mqiptGui
```

To allow the Administration Client to connect outwards through a firewall to an MQIPT, specify the host name or address and port number:

```
mqiptGui <socksHostName <socksPort>>
```

The default socksPort is 1080.

The status of the Administration Client is shown by messages appearing in the Administration Client’s main window.

Uninstalling internet pass-thru

Before uninstalling MQIPT from your system, prevent it from starting automatically, as described in “Starting internet pass-thru automatically”. Log in as root and run the installp command:

```
installp -u mqipt-RT
```

Chapter 16. Installing internet pass-thru on HP-UX

This chapter describes how you install MQIPT on an HP-UX system:

- “Downloading and installing the files”
- “Setting up internet pass-thru” on page 60
- “Starting internet pass-thru from the command line” on page 60
- “Starting internet pass-thru automatically” on page 61
- “Starting the Administration Client from the command line” on page 61
- “Uninstalling internet pass-thru” on page 62

Downloading and installing the files

MQIPT can be downloaded from the WebSphere MQ SupportPac Web page, at:

`http://www.ibm.com/webspheremq/supportpacs`

Follow the instructions for downloading.

Log in as `root`, uncompress and unpack `ms81_hp11.tar.Z` into a temporary directory. Run the `swinstall` command, as in this example:

```
login root
cd /tmp
uncompress -fv ms81_hp11.tar.Z
tar xvf ms81_hp11.tar
swinstall -s /tmp MQIPT.MQIPT-RT
```

The example assumes that `ms81_hp11.tar.Z` is in the /tmp directory.

MQIPT contains the files shown in the following table, including the files for the Administration Client GUI.

File	Purpose
Readme.txt	Latest information not included in the publications
mqiptSample.conf	Sample configuration file
ssl/sslSample.pfx	Test key ring file
ssl/sslSample.pwd	Password file for test key ring file
ssl/sslCAdefault.pfx	Sample Certificate Authority (CA) key ring file
ssl/sslCAdefault.pwd	Password file for sample CA key ring file
ssl/KeyMan.zip	KeyMan utility
exits/ SampleOneRouteExit.java	Sample security exit
exits/ SampleOneRouteExit.conf	Configuration file for SampleOneRouteExit
exits/SampleRoutingExit.java	Sample security exit
exits/SampleRoutingExit.conf	Configuration file for SampleRoutingExit
exits/SampleSecurityExit.java	Sample security exit
lib/MQipt.jar	Contains runtime, class, and property files
lib/ADV_mqipt_normal. class	Network Dispatcher advisor for “normal” mode

File	Purpose
lib/ADV_mqipt_replace. class	Network Dispatcher advisor for “replace” mode
lib/mqipt1414Sample.ssl	Sample trigger file for Network Dispatcher advisor
bin/mqipt	Shortcut for running MQIPT from the command line
bin/mqiptAdmin	Shortcut for stopping MQIPT and refreshing file information
bin/mqiptPW	Encrypting password used to open key ring files
bin/mqiptVersion	Display the version number of MQIPT
bin/mqiptService	For installing MQIPT so that it starts automatically at system startup.
bin/mqiptEnv	Defines the location of the `mqipt.jar` file and is used only by the other scripts.
bin/mqiptFork	Used to launch MQIPT during system startup
web/MQIPTServlet.war	Web archive file for servlet version
doc/<lang>/html/<filename>.zip	Master file for the *internet pass-thru* manual in HTML format. See “Bibliography” on page 169 for more information about softcopy documentation.
lib/guiadmin.jar	Contains runtime, class and property files for the Administration Client GUI
bin/mqiptGui	Shortcut for running the Administration Client GUI from the command line
bin/customSample. properties	Sample file for customizing the appearance and, therefore, accessibility of the Administration Client

Setting up internet pass-thru

Before starting MQIPT for the first time, copy the sample configuration file, `mqiptSample.conf`, to `mqipt.conf`. See Chapter 19, “Administering and configuring internet pass-thru”, on page 71 for further information.

Starting internet pass-thru from the command line

Log in as `root` and change directory to the `bin` directory. For example:

```
cd /opt/mqipt/bin
mqipt ..
```

Running the `mqipt` script without any options uses a default location of “.” for the configuration file (`mqipt.conf`). To specify a different location:

```
mqipt <directory name>
```

Messages will appear on the console showing the status of MQIPT. If an error occurs, see “Problem determination” on page 145. The following messages are an example of MQIPT successfully starting:

```
5639-L92 (C) Copyright IBM Corp. 2000, 2003 All Rights Reserved
MQCPI001 WebSphere MQ internet pass-thru Version 1.3.0 starting
MQCPI004 Reading configuration information from /opt/mqipt/mqipt.conf
MQCPI008 Listening for control commands on port 1881
MQCPI011 The path /opt/mqipt/logs will be used to store the log files
MQCPI006 Route 1418 has started and will forward messages to :
MQCPI034 ....mqserver.company4.com(1414)
MQCPI035 ....using MQ protocols
MQCPI078 Route 1418 ready for connection requests
```

```
MQCPI006 Route 1415 has started and will forward messages to :
MQCPI034 ....mqipt.company2.com(1415)
MQCPI035 ....using MQ protocols
MQCPI036 ....SSL Client side enabled with properties :
MQCPI031 ......cipher suites <null>
MQCPI032 ......keyring file /opt/mqipt/KeyMan.pfx
MQCPI038 ......distinguished name(s) CN=*Doe O=IBM OU=* L=* ST=* C=*
MQCPI078 Route 1415 ready for connection requests
```

The following subdirectories of the mqipt home directory are created automatically when MQIPT is invoked for the first time:

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

Starting internet pass-thru automatically

To start MQIPT automatically when the system is started, run the mqiptService script. For example:

```
cd /opt/mqipt/bin
mqiptService -install
```

This assumes that JDK 1.4 is already installed in a directory called /opt/java1.4. If this is not the case, edit file mqipt.ske and change the PATH variable to point to the location of the JDK. You must apply this change before running the mqiptService -install command.

When MQIPT is started as a service, it writes a console.log file to the logs subdirectory. This subdirectory is created the first time MQIPT is run, so MQIPT must be started at least once before trying to start it as a service.

To prevent MQIPT from starting automatically:

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

Starting the Administration Client from the command line

Open a command prompt and change directory to the bin directory and run mqiptGui. For example:

```
cd /opt/mqipt/bin
mqiptGui
```

To allow the Administration Client to connect outwards through a firewall to an MQIPT, specify the host name or address and port number:

```
mqiptGui <socksHostName <socksPort>>
```

The default socksPort is 1080.

The status of the Administration Client is shown by messages appearing in the Administration Client’s main window.

Uninstalling internet pass-thru

Before uninstalling MQIPT from your system, prevent it from starting automatically, as described in “Starting internet pass-thru automatically” on page 61. Log in as `root` and run the `swremove` command:

```
swremove MQIPT
```

Chapter 17. Installing internet pass-thru on Linux

This chapter describes how you install MQIPT on a Linux system:

- "Downloading and installing the files"
- "Setting up internet pass-thru" on page 64
- "Starting internet pass-thru from the command line" on page 64
- "Starting internet pass-thru automatically" on page 65
- "Starting the Administration Client from the command line" on page 65
- "Uninstalling internet pass-thru" on page 65

Downloading and installing the files

MQIPT can be downloaded from the WebSphere MQ SupportPac Web page, at:

http://www.ibm.com/webspheremq/supportpacs

Follow the instructions for downloading.

Log in as root, uncompress and unpack ms81_linux.tar.z into a temporary directory. Run the rpm command, as in this example:

```
login root
cd /tmp
uncompress -fv ms81_linux.tar.z
tar xvf ms81_linux.tar
cd i386
rpm -i WebSphereMQ-IPT-1.3.0-0.i386.rpm
```

The example assumes that ms81_linux.tar.z is in the /tmp directory.

MQIPT contains the files shown in the following table, including the files for the Administration Client GUI.

File	Purpose
Readme.txt	Latest information not included in the publications
mqiptSample.conf	Sample configuration file
ssl/sslSample.pfx	Test key ring file
ssl/sslSample.pwd	Password file for test key ring file
ssl/sslCAdefault.pfx	Sample Certificate Authority (CA) key ring file
ssl/sslCAdefault.pwd	Password file for sample CA key ring file
ssl/KeyMan.zip	KeyMan utility
exits/ SampleOneRouteExit.java	Sample security exit
exits/ SampleOneRouteExit.conf	Configuration file for SampleOneRouteExit
exits/SampleRoutingExit.java	Sample security exit
exits/SampleRoutingExit.conf	Configuration file for SampleRoutingExit
exits/SampleSecurityExit.java	Sample security exit
lib/libmqiptqos.so	Dummy library for TQoS

File	Purpose
bin/mqiptQoS	For using the real TQoS library
lib/MQipt.jar	Contains runtime, class, and property files
lib/ADV_mqipt_normal. class	Network Dispatcher advisor for “normal” mode
lib/ADV_mqipt_replace. class	Network Dispatcher advisor for “replace” mode
lib/mqipt1414Sample.ssl	Sample trigger file for Network Dispatcher advisor
lib/libiptqos.so	Runtime library for Quality of Service support
bin/mqipt	Shortcut for running MQIPT from the command line
bin/mqiptAdmin	Shortcut for stopping MQIPT and refreshing file information
bin/mqiptPW	Encrypting password used to open key ring files
bin/mqiptVersion	Display the version number of MQIPT
bin/mqiptService	For installing MQIPT so that it starts automatically at system startup.
bin/mqiptEnv	Defines the location of the mqipt.jar file and is used only by the other scripts.
web/MQIPTServlet.war	Web archive file for servlet version
doc/<lang>/html/<filename>.zip	Master file for the *internet pass-thru* manual in HTML format. See “Bibliography” on page 169 for more information about softcopy documentation.
lib/guiadmin.jar	Contains runtime, class and property files for the Administration Client GUI
bin/mqiptGui	Shortcut for running the Administration Client GUI from the command line
bin/customSample. properties	Sample file for customizing the appearance and, therefore, accessibility of the Administration Client

Setting up internet pass-thru

Before starting MQIPT for the first time, copy the sample configuration file, mqiptSample.conf, to mqipt.conf. See Chapter 19, “Administering and configuring internet pass-thru”, on page 71 for further information.

Starting internet pass-thru from the command line

Log in as root and change directory to the bin directory. For example:

```
cd /opt/mqipt/bin
mqipt ..
```

Running the mqipt script without any options uses a default location of “.” for the configuration file (mqipt.conf). To specify a different location:

```
mqipt <directory name>
```

Messages will appear on the console showing the status of MQIPT. If an error occurs, see “Problem determination” on page 145. The following messages are an example of MQIPT successfully starting:

```
5639-L92 (C) Copyright IBM Corp. 2000, 2003 All Rights Reserved
MQCPI001 WebSphere MQ internet pass-thru Version 1.3.0 starting
MQCPI004 Reading configuration information from /opt/mqipt/mqipt.conf
MQCPI008 Listening for control commands on port 1881
```

```
MQCPI011 The path /opt/mqipt/logs will be used to store the log files
MQCPI006 Route 1418 has started and will forward messages to :
MQCPI034 ....mqserver.company4.com(1414)
MQCPI035 ....using MQ protocols
MQCPI078 Route 1418 ready for connection requests
MQCPI006 Route 1415 has started and will forward messages to :
MQCPI034 ....mqipt.company2.com(1415)
MQCPI035 ....using MQ protocols
MQCPI036 ....SSL Client side enabled with properties :
MQCPI031 ......cipher suites <null>
MQCPI032 ......keyring file /opt/mqipt/KeyMan.pfx
MQCPI038 ......distinguished name(s) CN=*Doe O=IBM OU=* L=* ST=* C=*
MQCPI078 Route 1415 ready for connection requests
```

The following subdirectories of the mqipt home directory are created automatically when MQIPT is invoked for the first time:

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

Starting internet pass-thru automatically

To start MQIPT automatically when the system is started, run the mqiptService script. For example:

```
cd /opt/mqipt/bin
mqiptService -install
```

When MQIPT is started as a service, it writes a console.log file to the logs subdirectory. This subdirectory is created the first time MQIPT is run, so MQIPT must be started at least once before trying to start it as a service.

To prevent MQIPT from starting automatically:

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

Starting the Administration Client from the command line

Open a command prompt and change directory to the bin directory and run mqiptGui. For example:

```
cd /opt/mqipt/bin
mqiptGui
```

To allow the Administration Client to connect outwards through a firewall to an MQIPT, specify the host name or address and port number:

```
mqiptGui <socksHostName <socksPort>>
```

The default socksPort is 1080.

The status of the Administration Client is shown by messages appearing in the Administration Client’s main window.

Uninstalling internet pass-thru

Before uninstalling MQIPT from your system, prevent it from starting automatically, as described in “Starting internet pass-thru automatically”. Log in as root and run the swremove command:

```
rpm -e WebSphereMQ-IPT-1.3.0-0
```

Chapter 18. Generic UNIX installation

A disk image of all the common MQIPT files is provided in a tar file, for general use. The purpose of this file is to allow MQIPT to be install on those UNIX platforms not supported by MQIPT with their own install images. The intention is to allow the tar file to be unpacked into a specified location and with possibly minor changes, allow MQIPT to be implemented on any platform that supports Java 1.4. The mqiptEnv script, found in the bin subdirectory, may need to be changed to reflect the location of the installed files.

- "Downloading and installing the files"
- "Setting up internet pass-thru" on page 68
- "Starting internet pass-thru from the command line" on page 69
- "Starting internet pass-thru automatically" on page 69
- "Starting the Administration Client from the command line" on page 69
- "Uninstalling internet pass-thru" on page 70

Downloading and installing the files

MQIPT can be downloaded from the WebSphere MQ SupportPac Web page, at:

`http://www.ibm.com/webspheremq/supportpacs`

Follow the instructions for downloading.

Log in as root and unpack ms81.tar into the target directory, as in this example:

```
login root
cd /
mkdir mqipt
cd mqipt
cp /tmp/ms81.tar /mqipt/.
tar xvf ms81.tar
```

The example assumes that ms81.tar was downloaded to the /tmp directory.

MQIPT contains the files shown in the following table, including the files for the Administration Client GUI.

File	Purpose
Readme.txt	Latest information not included in the publications
mqiptSample.conf	Sample configuration file
ssl/sslSample.pfx	Test key ring file
ssl/sslSample.pwd	Password file for test key ring file
ssl/sslCAdefault.pfx	Sample Certificate Authority (CA) key ring file
ssl/sslCAdefault.pwd	Password file for sample CA key ring file
ssl/KeyMan.zip	KeyMan utility
exits/ SampleOneRouteExit.java	Sample security exit
exits/ SampleOneRouteExit.conf	Configuration file for SampleOneRouteExit

File	Purpose
exits/SampleRoutingExit.java	Sample security exit
exits/SampleRoutingExit.conf	Configuration file for SampleRoutingExit
exits/SampleSecurityExit.java	Sample security exit
lib/MQipt.jar	Contains runtime, class, and property files
lib/ADV_mqipt_normal. class	Network Dispatcher advisor for "normal" mode
lib/ADV_mqipt_replace. class	Network Dispatcher advisor for "replace" mode
lib/mqipt1414Sample.ssl	Sample trigger file for Network Dispatcher advisor
bin/mqipt	Shortcut for running MQIPT from the command line
bin/mqiptAdmin	Shortcut for stopping MQIPT and refreshing file information
bin/mqiptPW	Encrypting password used to open key ring files
bin/mqiptVersion	Display the version number of MQIPT
bin/mqiptService	For installing MQIPT so that it starts automatically at system startup.
bin/mqiptEnv	Defines the location of the `mqipt.jar` file and is used only by the other scripts.
web/MQIPTServlet.war	Web archive file for servlet version
doc/<lang>/html/ <filename>.zip	Master file for the *internet pass-thru* manual in HTML format. See "Bibliography" on page 169 for more information about softcopy documentation.
lib/guiadmin.jar	Contains runtime, class and property files for the Administration Client GUI
bin/mqiptGui	Shortcut for running the Administration Client from the command line
bin/customSample. properties	Sample file for customizing the appearance and, therefore, accessibility of the Administration Client

Setting up internet pass-thru

Before starting MQIPT for the first time, copy the sample configuration file, `mqiptSample.conf`, to `mqipt.conf`. See Chapter 19, "Administering and configuring internet pass-thru", on page 71 for further information.

This example assumes MQIPT will be unpacked into a directory called mqipt. You must update the mqiptEnv script with the new location of the runtime libraries. The default value of the MQIPT_CP variable is:

```
MQIPT_CP=/opt/mqipt/lib/MQipt.jar:/opt/mqipt/lib/guiadmin.jar
```

For our example, this must be changed to:

```
MQIPT_CP=/mqipt/opt/mqipt/lib/MQipt.jar:/mqipt/opt/mqipt/lib/guiadmin.jar
```

You must also update any of the runtime scripts before they are used and change the fully qualified path name for the location of the mqiptEnv script. So, for example, before using the mqipt script, edit it and change the statement after the comment `Get classpath` from:

```
/opt/mqipt/bin/mqiptEnv
```

to

```
/mqipt/opt/mqipt/bin/mqiptEnv
```

Starting internet pass-thru from the command line

Log in as `root` and change directory to the `bin` directory. For example:

```
cd /mqipt/opt/mqipt/bin
mqipt ..
```

Running the `mqipt` script without any options uses a default location of "." for the configuration file (`mqipt.conf`). To specify a different location:

```
mqipt <directory name>
```

Messages will appear on the console showing the status of MQIPT. If an error occurs, see "Problem determination" on page 145. The following messages are an example of MQIPT successfully starting:

```
5639-L92 (C) Copyright IBM Corp. 2000, 2003 All Rights Reserved
MQCPI001 WebSphere MQ internet pass-thru Version 1.3.0 starting
MQCPI004 Reading configuration information from /mqipt/opt/mqipt/mqipt.conf
MQCPI008 Listening for control commands on port 1881
MQCPI011 The path /mqipt/opt/mqipt/logs will be used to store the log files
MQCPI006 Route 1418 has started and will forward messages to :
MQCPI034 ....mqserver.company4.com(1414)
MQCPI035 ....using MQ protocols
MQCPI078 Route 1418 ready for connection requests
MQCPI006 Route 1415 has started and will forward messages to :
MQCPI034 ....mqipt.company2.com(1415)
MQCPI035 ....using MQ protocols
MQCPI036 ....SSL Client side enabled with properties :
MQCPI031 ......cipher suites <null>
MQCPI032 ......keyring file /mqipt/opt/mqipt/KeyMan.pfx
MQCPI038 ......distinguished name(s) CN=*Doe O=IBM OU=* L=* ST=* C=*
MQCPI078 Route 1415 ready for connection requests
```

The following subdirectories of the `mqipt` home directory are created automatically when MQIPT is invoked for the first time:

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

Starting internet pass-thru automatically

Starting a service automatically is platform specific. The mqiptService script is provided solely as an example of how it is done on a Sun Solaris system. Depending on the system requirements, it may be easier to use platform-specific utilities to install MQIPT as a system service.

Starting the Administration Client from the command line

Open a command prompt and change directory to the `bin` directory and run `mqiptGui`. For example:

```
cd /mqipt/opt/mqipt/bin
../mqiptGui
```

To allow the Administration Client to connect outwards through a firewall to an MQIPT, specify the host name or address and port number:

```
mqiptGui <socksHostName <socksPort>>
```

The default socksPort is 1080.

The status of the Administration Client is shown by messages appearing in the Administration Client's main window.

Uninstalling internet pass-thru

As MQIPT was not installed using a system installable image, it can be uninstalled 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.

Chapter 19. Administering and configuring internet pass-thru

You configure MQIPT by making changes to the configuration file `mqipt.conf`. Do this by using the Administration Client, which is the recommended way, or by using the editor of your choice. Both techniques are described here, with reference information relevant to both:

- "Using the internet pass-thru Administration Client"
- "Using internet pass-thru line mode commands" on page 75
- "Configuration reference information" on page 76

Using the internet pass-thru Administration Client

You can use the Administration Client to configure and update one or more MQIPTs. It displays global properties for an MQIPT and route-specific properties.

Note that the Administration Client does not prerequisite Java 1.4.

The only data stored locally to the Administration Client is the list of MQIPTs, in a file called `client.conf`. Global and route properties are always retrieved from the MQIPT before they are displayed in the Administration Client.

Starting the Administration Client

Start the Administration Client by using the `mqiptGui` script found in the `bin` subdirectory of MQIPT. See the installation chapter for each platform for information about starting the Administration Client.

The first time the Administration Client is started, a dialog box is displayed, prompting you for connection information to an MQIPT. The information required is:

MQIPT Name
A name used to describe this MQIPT. Although this information is not essential, you are recommended to supply it.

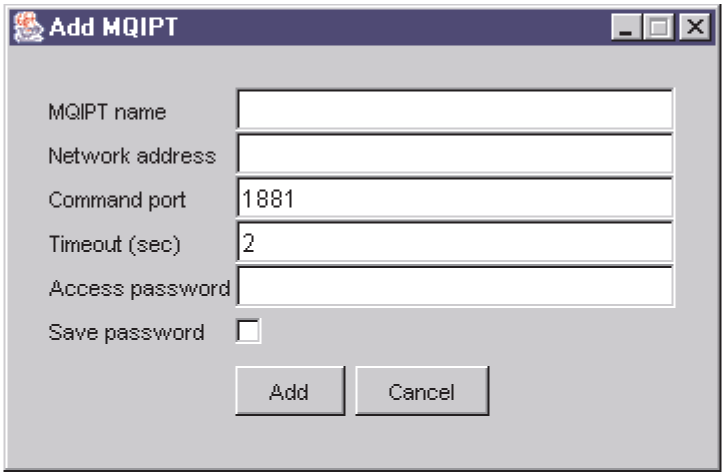
Network Address
The address of the system on which the MQIPT resides - either a name recognized by the name server, a dotted decimal address, or localhost (if the MQIPT is on the same machine as the client).

Command Port
The number of the port on which the MQIPT is listening for commands.

Timeout
This is the number of seconds the Administration Client will wait for a connection to the MQIPT. Keep this value as low as possible to reduce the refresh time of the window.

Access Password
The password used when communicating with the MQIPT. Fill in this field only if password checking is in force. (Password checking is in force if the AccessPW is provided in the MQIPT configuration file and is anything other than a null string.)

Save Password
If this checkbox is left blank, the password is remembered for the duration of the session, or until the MQIPT is removed. If the checkbox is selected, the password is saved for future sessions.



MQIPT name

Network address

Command port 1881

Timeout (sec) 2

Access password

Save password □

Figure 8. Window for first accessing an MQIPT

Administering an MQIPT

Only one MQIPT can be updated at a time, so, if another MQIPT is selected from the list, any outstanding changes must be applied before continuing. Changes made to any of the properties do not affect the MQIPT until the "Apply" menu option is used.

Selecting an MQIPT from the list retrieves the global and route properties from the MQIPT. If the MQIPT is not running or the incorrect CommandPort has been specified, an error message is issued. Changes to the host name and CommandPort can be made from the "Connection" menu option.

Double-clicking on an MQIPT from the list displays a list of routes. Selecting a route displays its properties. You can tailor the properties to your requirements.

When changes are applied, the configuration file is time stamped and sent back to the MQIPT and the changes take effect immediately. Any existing comment lines are lost.

A route can be added by using the "Add Route" menu option. A set of default properties is displayed for this new route, as defined by the global properties.

The inheritance of properties

There is a hierarchy of ways in which properties of MQIPTs and routes can be set in the Administration Client:

1. Every property has a default value and if the property is not mentioned in the configuration file, or has not been specifically set by user action in the Administration Client, this default value is assumed.
2. Global properties set on MQIPTs are assumed by every route on that MQIPT unless there is specific route information to the contrary. In the configuration file, this means that properties set in the global stanza are propagated to all routes unless additional properties are set in route stanzas. Properties set by the

Administration Client user on an MQIPT are propagated to all the routes unless a property is specifically set on a route.

3. Regardless of default values and global settings, any setting made against a route is sustained for that route.

File menu options

Most of the options relevant to managing the tree are shown when the File menu is selected.

Add MQIPT
Brings up the same dialog that appears when the client is first used, described in "Starting the Administration Client" on page 71.

Remove MQIPT
Removes the currently highlighted MQIPT only from the tree on the Administration Client. It does not affect the running of the MQIPT.

Save Configuration
Saves the MQIPT nodes of the tree to the Administration Client's configuration file so that they can be read back the next time it starts. Only the MQIPT nodes are saved. Global and route properties are always retrieved from the MQIPT.

Quit
Stops the Administration Client running. However, the Administration Client first checks whether the tree or the current MQIPT has changed; if either or both have, you are presented with a dialog or dialogs asking whether you wish to save the client, apply the changes to the MQIPT, or both.

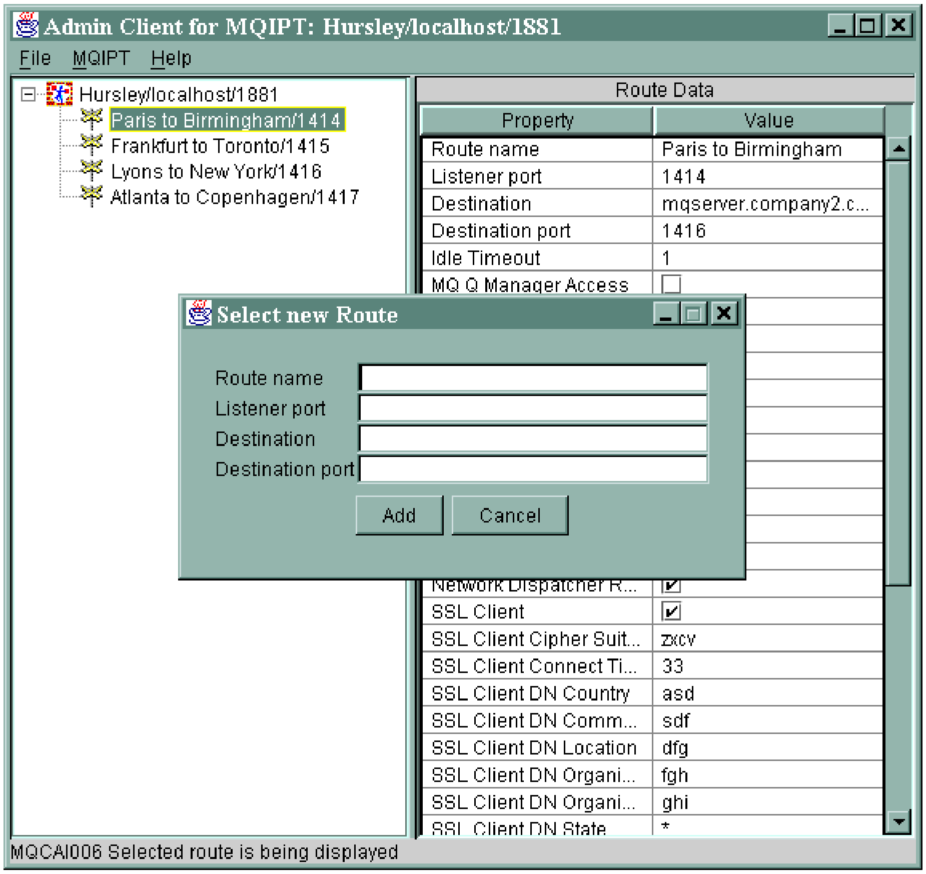
MQIPT menu options

Connection
Changes an MQIPT's access parameters. The changes are reflected in the tree view. It brings up a window similar to the one described in "Starting the Administration Client" on page 71.

Password
Changes the password property of the remote MQIPT. This action brings up a password dialog where you are expected to make the following entries:

- **Current Password**: as a check against improper use, you must demonstrate that you know the current password before you can change it. If no password is currently in force, this field is left blank.
- **New Password**: the new password or blank if you wish to discontinue the use of passwords on this MQIPT.
- **New Password Again**: protects you against typing mistakes in the previous field by asking you to repeat the same information.
- **Save Password**: used to determine whether the new password will be saved locally, along with the other access properties of this MQIPT.

Add Route
Adds a route to the selected MQIPT. See Figure 9 on page 74 for details. Each route must have a unique ListenerPort for the MQIPT.



Property	Value
Route name	Paris to Birmingham
Listener port	1414
Destination	mqserver.company2.c...
Destination port	1416
Idle Timeout	1
MQ Q Manager Access	☐
Network Dispatcher R...	☑
SSL Client	☑
SSL Client Cipher Suit...	zxcv
SSL Client Connect Ti...	33
SSL Client DN Country	asd
SSL Client DN Comm...	sdf
SSL Client DN Location	dfg
SSL Client DN Organi...	fgh
SSL Client DN Organi...	ghi
SSL Client DN State	*

MQCAI006 Selected route is being displayed

Figure 9. Adding a route

Delete Route
Deletes the selected route from the MQIPT. The deletion does not affect the MQIPT until the “Apply” menu option is used.

Apply
When you are satisfied with the changes you have made to the MQIPT’s configuration, this option sends a new configuration file to the MQIPT, which saves it. The new settings are made effective immediately.

Refresh
Reads the configuration file from the selected MQIPT and refreshes the display.

Stop
Sends a stop command to the MQIPT to tell it to stop running. After this command, you lose contact with the MQIPT. This command is ignored unless the global property RemoteShutdown is turned on.

Route information can be updated in the same way as MQIPT global information. When you change any properties of a route, you have to apply the changes before they take effect. You can do this either by selecting the “MQIPT/Apply” menu option or replying ″Yes″ when you are prompted about saving the configuration.

Help menu options

Help
Uses Netscape to display information on how to use the Administration Client,

select ″Administering and configuring internet pass-thru″ in the left hand pane. Before using the Administration Client you must unzip the files found in the <lang>/html subdirectory.

About
Shows a splash window with information about the version of the Administration Client.

Using internet pass-thru line mode commands

If you choose not to use the Administration Client, you can use line mode commands to administer and configure internet pass-thru.

Administering internet pass-thru using line mode commands

Using your editor of choice, change the configuration file, mqipt.conf, to meet your requirements. See “Configuration reference information” on page 76 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 the following ASCII administration commands:

```
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 not provided, hostname defaults to localhost and port to 1881.

STOP
MQIPT closes all connections, stops listening for incoming connections, and then exits. Using the “MQIPT/Stop” menu option of the Administration Client has the same effect. This command is ignored unless the mqipt.conf file specifies RemoteShutDown=true.

REFRESH
MQIPT rereads mqipt.conf. If it finds:

- That any of the routes currently active are now marked as inactive (or are missing altogether), it closes them down and stops listening for incoming connections on those routes.
- Any routes marked active in the configuration file that it does not currently have running, it starts them up.
- That the configuration parameters of a currently running route have changed, it 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’s settings.

On Windows, 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 control the actions of the MQIPT server. The file comprises a set of sections. There is one global section, and an additional 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 sections, some can appear only in the route sections, and some may appear both in route and global sections. If a property does appear in both route and global sections, the property value 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 keyword names (that is to say, any name/value pairs where the name is not one of the names defined in this document) are ignored. If a name/value pair appearing 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 name/value pair appearing 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` and `false`, any mixture of upper- and lower-case can be used.

Changes to any of the properties can be made by either editing the mqipt.conf file or by using the Administration Client GUI. To apply any changes, the administrator can issue a refresh command, either from the Administration Client GUI or by using the mqiptAdmin script.

Changes to certain properties will only cause a route to be restarted if other properties are already enabled. For example, any changes to the HTTP properties will only have an effect 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 on how to set up some simple configurations, see Chapter 20, “Getting started with internet pass-thru”, on page 93. For a sample configuration, see the `mqiptSample.conf` file in the home directory of MQIPT.

Summary of properties

Table 3 on page 77 shows:

- All the properties
- Whether the property applies to the global section, the route section, or both
- Default values used if a property is missing from both the route section and the global section

Table 3. Summary of configuration properties

Name of property	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
HTTP[6,7]	yes	yes	false
HTTPChunking[1]	yes	yes	false
HTTPProxy[1]	yes	yes	<null>
HTTPProxyPort[1]	yes	yes	8080
HTTPS[1]	yes	yes	false
HTTPServer[1]	yes	yes	<null>
HTTPServerPort[1]	yes	yes	<null>
IdleTimeout	yes	yes	0
IgnoreExpiredCRLs	yes	yes	false
LDAP	yes	yes	false
LDAPIgnoreErrors[10]	yes	yes	false
LDAPCacheTimeout[10]	yes	yes	24
LDAPSaveCRL[10]	yes	yes	false
LDAPServer1[10]	yes	yes	<null>
LDAPServer1Port[10]	yes	yes	389
LDAPServer1Userid[10]	yes	yes	<null>
LDAPServer1Password[10]	yes	yes	<null>
LDAPServer1Timeout[10]	yes	yes	0
LDAPServer2[10]	yes	yes	<null>
LDAPServer2Port[10]	yes	yes	389
LDAPServer2Userid[10]	yes	yes	<null>
LDAPServer2Password[10]	yes	yes	<null>
LDAPServer2Timeout[10]	yes	yes	0
ListenerPort	no	yes	<null>
LocalAddress	yes	yes	<null>
LogDir (this is only valid for MQIPTServlet)	no	no	<null>
MaxConnectionThreads	yes	yes	100
MaxLogFileSize	yes	no	50
MinConnectionThreads	yes	yes	5
Name	no	yes	<null>
NDAdvisor	yes	yes	false
NDAdvisorReplaceMode[4]	yes	yes	false
OutgoingPort	no	yes	0

Table 3. Summary of configuration properties (continued)

Name of property	Global	Route	Default
QMgrAccess	yes	yes	true
QoS (can only be used on Linux)	yes	yes	false
QosToCaller[9]	yes	yes	1
QosToDest[9]	yes	yes	1
RemoteShutdown	yes	no	false
RouteRestart	yes	yes	true
SecurityExit	yes	yes	false
SecurityExitName[11]	yes	yes	<null>
SecurityExitPath[11]	yes	yes	<ipthome> \exits
SecurityExitTimeout[11]	yes	yes	5
SecurityManager	yes	no	false
SecurityManagerPolicy	yes	no	<null>
ServletClient[1]	yes	yes	false
SocksClient	yes	yes	false
SocksProxyHost[8]	yes	yes	<null>
SocksProxyPort[8]	yes	yes	1080
SocksServer[7]	yes	yes	false
SSLClient	yes	yes	false
SSLClientCAKeyRing[2]	yes	yes	<null>
SSLClientCAKeyRingPW[2]	yes	yes	<null>
SSLClientCipherSuites[2]	yes	yes	<null>
SSLClientConnectTimeout[2]	yes	yes	30
SSLClientDN_C[2]	yes	yes	"*"[5]
SSLClientDN_CN[2]	yes	yes	"*"[5]
SSLClientDN_L[2]	yes	yes	"*"[5]
SSLClientDN_O[2]	yes	yes	"*"[5]
SSLClientDN_OU[2]	yes	yes	"*"[5]
SSLClientDN_ST[2]	yes	yes	"*"[5]
SSLClientKeyRing[2]	yes	yes	<null>
SSLClientKeyRingPW[2]	yes	yes	<null>
SSLClientSiteDN_C[2]	yes	yes	"*"[5]
SSLClientSiteDN_CN[2]	yes	yes	"*"[5]
SSLClientSiteDN_L[2]	yes	yes	"*"[5]
SSLClientSiteDN_O[2]	yes	yes	"*"[5]
SSLClientSiteDN_OU[2]	yes	yes	"*"[5]
SSLClientSiteDN_ST[2]	yes	yes	"*"[5]
SSLClientSiteLabel[2]	yes	yes	<null>
SSLProxyMode	yes	yes	false
SSLServer[6]	yes	yes	false

Table 3. Summary of configuration properties (continued)

Name of property	Global	Route	Default
SSLServerAskClientAuth[3]	yes	yes	false
SSLServerCAKeyRing[3]	yes	yes	<null>
SSLServerCAKeyRingPW[3]	yes	yes	<null>
SSLServerCipherSuites[3]	yes	yes	<null>
SSLServerDN_C[3]	yes	yes	"*"[5]
SSLServerDN_CN[3]	yes	yes	"*"[5]
SSLServerDN_L[3]	yes	yes	"*"[5]
SSLServerDN_O[3]	yes	yes	"*"[5]
SSLServerDN_OU[3]	yes	yes	"*"[5]
SSLServerDN_ST[3]	yes	yes	"*"[5]
SSLServerKeyRing[3]	yes	yes	<null>
SSLServerKeyRingPW[3]	yes	yes	<null>
SSLServerSiteDN_C[3]	yes	yes	"*"[5]
SSLServerSiteDN_CN[3]	yes	yes	"*"[5]
SSLServerSiteDN_L[3]	yes	yes	"*"[5]
SSLServerSiteDN_O[3]	yes	yes	"*"[5]
SSLServerSiteDN_OU[3]	yes	yes	"*"[5]
SSLServerSiteDN_ST[3]	yes	yes	"*"[5]
SSLServerSiteLabel[3]	yes	yes	<null>
Trace	yes	yes	0
UriName (See page “UriName” on page 92 for details about the default settings.)[1]	yes	yes	

Notes:

1. Set HTTP to true for these properties to have an effect.
2. Set SSLClient to true for these properties to have an effect.
3. Set SSLServer to true for these properties to have an effect.
4. Set NDAdvisor to true for these properties to have an effect.
5. The "*" symbol represents a wildcard.
6. HTTP and SSLServer cannot be used together. The HTTP property is only used to define the forward connection. Incoming data on the ListenerPort is detected automatically, setting SSLServer will cause a runtime exception.
7. HTTP and SocksServer cannot be used together. The HTTP property is only used to define the forward connection. Incoming data on the ListenerPort is detected automatically, setting SocksServer will cause a runtime exception.
8. Set SocksClient to true for these properties to have an effect.
9. Set QoS to true for these properties to have an effect.
10. Set LDAP to true for these properties to have an effect.
11. Set SecurityExit to true for these properties to have an effect.

Global section reference information

The global section may contain the following properties and all the properties in "Route section reference information", apart from ListenerPort, Destination, DestinationPort, Name and OutgoingPort.

AccessPW
The password used when an Administration Controller 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. If you want to listen on the command port, you are advised to use 1881. The Administration Client does not have a default value for CommandPort, but 1881 is the default value when you use line mode commands.

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 `mqiptYYYYMMDDHHmmSS.log`. The default value is true. When this property is changed from `true` to `false`, MQIPT closes the existing connection log and creates a new one. The new one will be 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 (`mqipt.back`) is made, and a new file is started. Only one backup file is kept; each time the main log file fills up, any earlier backups are erased. The default value 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. This relies on the correct permissions being granted. See "Java Security Manager" on page 31 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 just 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.

Route section reference information

The route section may 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 `Active=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 this route are terminated.

ClientAccess

The route allows incoming client channel connections only if the value of ClientAccess is set to `true`. Note that potentially you can configure MQIPTs 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 this route are terminated.

Destination

The hostname (or dotted decimal IP address) of the queue manager (or subsequent MQIPT) to which this route is to connect. Each route section **must** contain an explicit Destination value. You are allowed to have several route sections pointing 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 this route are terminated.

DestinationPort

The port on the Destination host to which this route is to connect. It is valid for more than one route to point at the same combination of Destination and DestinationPort. Each route section **must** contain an explicit DestinationPort value. If a change to this property affects a route, the route is stopped and restarted when a REFRESH command is issued. All connections to this route are terminated.

HTTP

Set this to `true` for routes responsible for making outbound HTTP tunneling requests (that is, communicating with another MQIPT over HTTP). Set to `false` for routes directed at WebSphere MQ queue managers. If you change this property, the route is stopped and restarted when a REFRESH command is issued. All connections to this route are terminated. To use HTTP chunking, set this property to `true`. This property cannot be used with:

- QoS
- SocksClient
- SSLClient
- SSLProxyMode

HTTPChunking

Set this to `true` for routes responsible for making outbound requests using HTTP tunneling with chunking. The HTTP property must also be set to `true`. Set to `false` when you are not using HTTP chunking. 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 this route are terminated.

HTTPProxy

The host name (or dotted decimal IP address) of the HTTP proxy that all connections for this route use. If HTTPServer is also defined, then a CONNECT request is issued to the HTTPProxy, instead of a normal POST. 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 this route are terminated.

HTTPProxyPort
The port address to use on the HTTP proxy. The default value is `8080`, unless HTTPS has been set to true and there is no HTTPServer, and then 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 this route are terminated.

HTTPServer
The host name (or dotted decimal IP address) of the HTTP server that all connections for this route use. 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 this route are terminated.

HTTPS
Enable this property to make HTTPS requests. The HTTP property must also be enabled. 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 this route are terminated.

HTTPServerPort
The port address to use on the HTTP server. The default value is `8080`, unless HTTPS has been set to true and then 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 this route are terminated.

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. A value of 0 indicates no idle timeout. Changes to this property take effect only when the route is restarted.

IgnoreExpiredCRLs
Set this property to true to ignore an expired CRL. The default value is false.

> **Attention**
> If you enable this property, a revoked certificate could be used to make an SSL connection.

LDAP
Set this property to true to enable use of an LDAP server when using SSL connections. MQIPT will use the LDAP server to retrieve CRLs and ARLs. The SSLClient or SSLServer property must also be enabled for this property to have an effect.

LDAPIgnoreErrors
Set this property 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 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 this route are terminated.

Attention

If you enable this property, a revoked certificate could be used to make an SSL connection.

LDAPCacheTimeout
When a CRL has been retrieved from an LDAP server, it will be stored internally to MQIPT in a temporary cache. Entries in this cache will expire after a certain timeout, specified by this property. The default value is 24 hours. Specifying a timeout value of 0 means that 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 this route are terminated.

LDAPSaveCRL
Set this property to true to update the given key ring file with any CRLs retrieved from the LDAP server. Key ring files are specified with the SSLClientKeyRing, SSLClientCAKeyRing, SSLServerKeyRing and SSLServerCAKeyRing properties. This implies MQIPT must have write access to the key ring files. 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 terminated.

LDAPServer1
Set this property to the hostname or IP address of the main LDAP server. This property must be set if LDAP has been enabled. 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 terminated.

LDAPServer1Port
Set this property to the listening port address of the main LDAP server. It has a default value of 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 this route are terminated.

LDAPServer1Userid
Set this property to the userid 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 this route are terminated.

LDAPServer1Password
Set this property to the password needed to access the main LDAP server. This property must be set if LDAPServer1Userid has been set. 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 terminated.

LDAPServer1Timeout
Set this property to the number of seconds MQIPT will wait for a response from the main LDAP server. It has a default value of 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 this route are terminated.

LDAPServer2
Set this property to the hostname 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 this route are terminated.

LDAPServer2Port
Set this property to the listening port address of the backup LDAP server. It has a default value of 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 this route are terminated.

LDAPServer2Userid
Set this property to 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 this route are terminated.

LDAPServer2Password
Set this property to the password needed to access the backup LDAP server. This property must be set if LDAPServer2 has been enabled. 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 terminated.

LDAPServer2Timeout
Set this property to the number of seconds MQIPT will wait for a response from the backup LDAP server. It has a default value of 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 this route are terminated.

ListenerPort
The port number on which the route should listen for incoming requests. Each route section **must** contain an explicit ListenerPort value; moreover, 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 local IP address to bind all connections to. If you change this property, the route is stopped and restarted when a REFRESH command is issued. All connections to this route are terminated.

LogDir
Use this property to define the directory name for log and trace files. Changes to this property will not be effected until MQIPTServlet has been stopped and restarted. The default value is <null>. This property is only valid for MQIPTServlet

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 will be queued once all the threads are in use. Beyond that number, subsequent connection requests are refused. The minimum allowed value is the greater of 1 or 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 pick up the new value immediately. The route is not terminated.

MinConnectionThreads
The minimum number of connection threads (threads to handle incoming

connections on this route). This is the number of threads allocated when the route is started, and the total number of threads allocated does not drop below this value during the time the route is active. The minimum allowed value is 0 and the value must be less than that specified for MaxConnectionThreads. Changes to this property take effect only when the route is restarted.

Name
An optional name to help identify the route. It appears in console messages and tracing information. Changes to this property take effect only when the route is restarted.

NDAdvisor
Set this property 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 this route are terminated. To use the NDAdvisorReplaceMode property, set this property to `true`.

NDAdvisorReplaceMode
Set this property to `true` to use the "replace" mode of the Network Dispatcher custom advisor. You must have started the `mqipt_replace custom advisor` for the ListenerPort address of this route. Set this property to `false` to use "normal" mode. You must set the NDAdvisor property to `true` to use this property.

OutgoingPort
This is the starting port address used by outgoing connections. The range of port addresses match the `MaxConnectionThread` value for this route. A default value of `0` will use a system defined port address. If you change this property, the route is stopped and restarted when a REFRESH command is issued. All connections to this route are terminated.

QMgrAccess
The route allows incoming queue manager channel connections (for example sender channels) only if the value of QMgrAccess is set to the value `true`. If you change this property to `false`, the route is stopped when a REFRESH command is issued. All connections to this route are terminated.

QoS
Set this property to `true` to enable Quality of Service for all connections on this route. This property can only be enabled on Linux. If you change this property, the route is stopped and restarted when a REFRESH command is issued. All connections to this route are terminated. This property cannot be used with:

- HTTP
- SSLClient
- SSLProxyMode
- SSLServer

QosToCaller
This property sets the priority of all traffic from the MQIPT machine to the initiator of the connection. For example, set the property to 1 for low priority, 2 for medium priority, and 3 for high priority (the default is 1). If you change this property (and QoS is set to `true`), the route is stopped and restarted when a REFRESH command is issued. All connections to this route are terminated

QosToDest
This property sets the priority of all traffic from the MQIPT machine to the destination of the connection (as defined by the Destination property). For example, set the property to 1 for low priority, 2 for medium priority, and 3 for

high priority (the default is 1). If you change this property (and QoS is set to `true`), the route is stopped and restarted when a REFRESH command is issued. All connections to this route are terminated

RouteRestart
Set this property 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 this property 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 terminated.

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 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 terminated.

SecurityExitTimeout
This timeout value is used by MQIPT to determine how long to wait (in seconds) for a response when validating a connection request. The default value is 5 seconds. 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 terminated.

ServletClient
Set this property to `true` when connecting to the MQIPT servlet. The HTTP property must also be set to `true`. If you change this property (and HTTP is set to `true`) the route is stopped and restarted when a REFRESH command is issued.

SocksClient
Set this property 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 this route are terminated. 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 terminated

SocksProxyPort
The port address 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 this route are terminated

SocksServer

Set this property 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 this route are terminated. This property cannot be used with:

- SocksClient
- SSLProxyMode
- SSLServer

SSLClient

Set this property to `true` to make the route act as an SSL client and make outgoing SSL connections. Setting `true` implies that the destination is either another MQIPT acting as an SSL server or an HTTP proxy/server. You must specify the name of a key ring file either with the SSLClientKeyRing or SSLClientCAKeyRing property. If you change this property, the route is stopped and restarted when a REFRESH command is issued. All connections to this route are terminated. This property cannot be used with:

- HTTP
- QoS
- SSLProxyMode

SSLClientCAKeyRing

The fully-qualified file name of the key ring file containing CA certificates, used to authenticate certificates from the SSL server. On Windows platforms, you must use a double back slash (\\) 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 terminated.

SSLClientCAKeyRingPW

The fully-qualified file name containing the password to open the client CA key ring. On Windows platforms, you must use a double back slash (\\) 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 terminated.

SSLClientCipherSuites

The name of the SSL cipher suite to use on the SSL client side. This can be one or more of the supported cipher suites. If you leave this blank, the SSL client uses the supported cipher suites 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 terminated.

SSLClientConnectTimeout

Set this property to the number of seconds an SSL client will wait for an SSL 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 this route are terminated.

SSLClientDN_C

Use this property to accept certificates received from the SSL server of this country name. The name can be prefixed or suffixed with an asterisk (*) to extend its scope. If you do not specify this property, you imply “all country names". 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 terminated.

SSLClientDN_CN
Use this property to accept certificates received from the SSL server of this common name. The name can be prefixed or suffixed with an asterisk (*) to extend its scope. If you do not specify this property, you imply "all country names". 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 terminated.

SSLClientDN_L
Use this property to accept certificates received from the SSL server of this location. The name can be prefixed or suffixed with an asterisk (*) to extend its scope. 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 this route are terminated.

SSLClientDN_O
Use this property to accept certificates received from the SSL server of this organization. The name can be prefixed or suffixed with an asterisk (*) to extend its scope. If you do not specify this property, you imply "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 this route are terminated.

SSLClientDN_OU
Use this property to accept certificates received from the SSL server of this organizational unit. The name can be prefixed or suffixed with an asterisk (*) to extend its scope. If you do not specify this property, you imply "all organizational units". 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 terminated.

SSLClientDN_ST
Use this property to accept certificates received from the SSL server of this state. The name can be prefixed or suffixed with an asterisk (*) to extend its scope. If you do not specify this property, you imply "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 this route are terminated.

SSLClientKeyRing
The fully-qualified file name of the key ring file containing the client certificate. On **Windows platforms**, you must use a double back slash (\\) as the file separator. You must specify SSLClientKeyRing if you set SSLClient to `true`. 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 terminated.

SSLClientKeyRingPW
The fully-qualified file name containing the password to open the client key ring. On **Windows platforms**, you must use a double back slash (\\) as the file separator. You must specify SSLClientKeyRingPW if you set SSLClient to `true`. 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 terminated.

SSLClientSiteDN_C

Use this property to specify a Country name to select a certificate to send to the SSL server. If you do not specify this property, you imply ″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 this route are terminated.

SSLClientSiteDN_CN

Use this property to specify a Common Name to select a certificate to send to the SSL server. If you do not specify this property, you imply ″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 this route are terminated.

SSLClientSiteDN_L

Use this property to specify a Location name to select a certificate to send to the SSL server. If you do not specify this property, you imply ″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 this route are terminated.

SSLClientSiteDN_O

Use this property to specify an Organization name to select a certificate to send to the SSL server. If you do not specify this property, you imply ″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 this route are terminated.

SSLClientSiteDN_OU

Use this property to specify an Organizational Unit name to select a certificate to send to the SSL server. If you do not specify this property, you imply ″any organizational unit 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 terminated.

SSLClientSiteDN_ST

Use this property to specify a State name to select a certificate to send to the SSL server. If you do not specify this property, you imply ″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 this route are terminated.

SSLClientSiteLabel

Use this property to specify a label name to select a certificate to send to the SSL server. If you do not specify this property, you imply ″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 this route are terminated.

SSLProxyMode

Set this property to `true` to enable the route to only accept SSL client connection requests and 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 terminated. This property cannot be used with:

- HTTP
- QoS
- SocksClient

- SSLClient
- SSLServer

SSLServer
Set this property to `true` to make the route act as an SSL server and accept incoming SSL connections. Setting `true` implies that the caller is another MQIPT acting as an SSL client. If you change this property, the route is stopped and restarted when a REFRESH command is issued. All connections to this route are terminated. This property cannot be used with:

- QoS
- SocksServer
- SSLProxyMode

SSLServerCAKeyRing
The fully-qualified file name of the key ring file containing CA certificates, used to authenticate certificates from the SSL client. On Windows platforms, you must use a double back slash (\\) 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 terminated.

SSLServerCAKeyRingPW
The fully-qualified file name containing the password to open the server CA key ring. On Windows platforms, you must use a double back slash (\\) 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 terminated.

SSLServerAskClientAuth
Use this property to request SSL client authentication by the SSL server. The SSL client must have its own certificate to send to the SSL 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 terminated.

SSLServerCipherSuites
The name of the SSL cipher suite to use on the SSL server side. This can be one or more of the supported cipher suites. If you leave this blank, the SSL server uses the supported cipher suites from the SSLServerKeyRing. 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 terminated.

SSLServerDN_C
Use this property to accept certificates received from the SSL client of this country name. The name can be prefixed or suffixed with an asterisk (*) to extend its scope. If you do not specify this property, you imply "all company names". 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 terminated.

SSLServerDN_CN
Use this property to accept certificates received from the SSL client of this common name. The name can be prefixed or suffixed with an asterisk (*) to extend its scope. If you do not specify this property, you imply "all common names". 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 terminated.

SSLServerDN_L

Use this property to accept certificates received from the SSL client of this location. The name can be prefixed or suffixed with an asterisk (*) to extend its scope. If you do not specify this property, you imply “all locations”. 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 terminated.

SSLServerDN_O

Use this property to accept certificates received from the SSL client of this organization. The name can be prefixed or suffixed with an asterisk (*) to extend its scope. If you do not specify this property, you imply “all organizations”. 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 terminated.

SSLServerDN_OU

Use this property to accept certificates received from the SSL client of this organizational unit. The name can be prefixed or suffixed with an asterisk (*) to extend its scope. If you do not specify this property, you imply “all organizational units”. 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 terminated.

SSLServerDN_ST

Use this property to accept certificates received from the SSL client of this state. The name can be prefixed or suffixed with an asterisk (*) to extend its scope. If you do not specify this property, you imply “all states”. 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 terminated.

SSLServerKeyRing

The fully-qualified file name of the key ring file containing the server certificate. On **Windows platforms**, you must use a double back slash (\\) as the file separator. You must specify SSLServerKeyRing if you set SSLServer to `true`. If you change this property (and SSLServer is set to true), the route is stopped and restarted when a REFRESH command is issued. All connections to this route are terminated.

SSLServerKeyRingPW

The fully-qualified file name containing the password to open the server key ring. On **Windows platforms**, you must use a double back slash (\\) as the file separator. You must specify SSLServerKeyRingPW if you set SSLServer to `true`. If you change this property (and SSLServer is set to true), the route is stopped and restarted when a REFRESH command is issued. All connections to this route are terminated.

SSLServerSiteDN_C

Use this property to specify a Country name to select a certificate to send to the SSL client. If you do not specify this property, you imply ″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 this route are terminated.

SSLServerSiteDN_CN

Use this property to specify a Common Name to select a certificate to send to the SSL client. If you do not specify this property, you imply ″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 this route are terminated.

SSLServerSiteDN_L
Use this property to specify a Location name to select a certificate to send to the SSL client. If you do not specify this property, you imply ″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 this route are terminated.

SSLServerSiteDN_O
Use this property to specify an Organization name to select a certificate to send to the SSL client. If you do not specify this property, you imply ″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 this route are terminated.

SSLServerSiteDN_OU
Use this property to specify an Organizational Unit name to select a certificate to send to the SSL client. If you do not specify this property, you imply ″any organizational unit 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 terminated.

SSLServerSiteDN_ST
Use this property to specify a State name to select a certificate to send to the SSL client. If you do not specify this property, you imply ″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 this route are terminated.

SSLServerSiteLabel
Use this property to specify a label name to select a certificate to send to the SSL client. If you do not specify this property, you imply ″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 this route are terminated.

Trace
The level of tracing required can be specified by an integer in the range 0-5. A value of 0 means no tracing; 5 requests full tracing.

If a change to this property affects a route, the new value is used when the REFRESH command is issued. All connections pick up the new value immediately. The route is not terminated.

UriName
This property can be used to changed the name of the Uniform Resource Identifier of the resource when using an HTTP proxy or the MQIPT servlet, although the default values will suffice for most configurations. The default for HTTP proxy is:

```
HTTP://<destination>:<destination_port>/mqipt
```

The default for the MQIPT servlet is:

```
HTTP://<destination>:<destination_port>/MQIPTServlet
```

If you change this property (and HTTP or ServletClient are set to `True`, the route is stopped and restarted when a REFRESH command is issued.

Chapter 20. Getting started with internet pass-thru

This chapter helps you get started with MQIPT: it takes you through the setup of some simple configurations to confirm that the product has installed successfully.

This chapter has the following sections:
- “Assumptions”
- “Example configurations” on page 94
- “Installation Verification Test” on page 94
- “SSL server authentication” on page 96
- “SSL client authentication” on page 98
- “HTTP proxy configuration” on page 101
- “Configuring access control” on page 103
- “Configuring Quality of Service (QoS)” on page 105
- “Configuring SOCKS proxy” on page 109
- “Configuring SOCKS client” on page 111
- “Creating SSL test certificates” on page 112
- “Configuring the MQIPT Servlet” on page 113
- “HTTPS configuration” on page 116
- “Configuring MQIPT Clustering support” on page 119
- “Creating a key ring file” on page 123
- “Allocating port addresses” on page 125
- “Using an LDAP server” on page 126
- “SSL proxy mode” on page 130
- “Apache rewrite” on page 132
- “Security exit” on page 136
- “Routing security exit” on page 138
- “Dynamic one route exit” on page 141

Assumptions

For each example, we make the following assumptions:
- You are using Windows NT, (although these examples will run on any of the supported platforms)
- You are familiar with defining queue managers, queues, and channels on WebSphere MQ
- You have already installed a WebSphere MQ client and server
- MQIPT is installed in a directory called `C:\mqipt` (on Windows)
- The client, server, and each MQIPT are installed on separate machines
- You are familiar with putting messages on a queue using the `amqsputc` command
- You are familiar with getting messages from a queue using the `amqsgetc` command

On the WebSphere MQ server you have done the following:
- Defined a queue manager called `MQIPT.QM1`

- Defined a server connection channel called MQIPT.CONN.CHANNEL
- Defined a local queue called MQIPT.LOCAL.QUEUE
- Started a TCP/IP listener for MQIPT.QM1 on port 1414

Only one application can listen on a given port address on the same machine. If port 1414 is already in use, choose a free port address and substitute it in the following examples.

Once you have done this you can test the route from the WebSphere MQ Client to the queue manager by putting a message on the local queue of the queue manager using the amqsputc command and retrieving it using the amqsgetc command.

Example configurations

The following examples are represented as diagrams and step-by-step instructions, you can use the tick boxes on the right hand side of each diagram to track your progress through the example. In some of the examples you are required to edit the mqipt.conf file, this can be found in the MQIPT home directory.

Before you begin, ensure that you have done the following:

- Copy mqiptSample.conf to mqipt.conf
- Edit mqipt.conf and delete all routes
- Change the entry for ClientAccess to True
- Change the Destination from mqserver.company2.com to that of your queue manager
- Change the DestinationPort address to that used by your queue manager
- Read “Assumptions” on page 93

Installation Verification Test

This is a simple configuration to ensure that MQIPT has installed correctly.

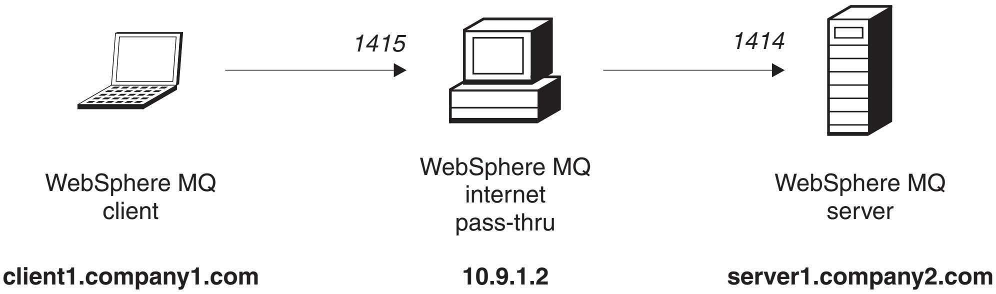
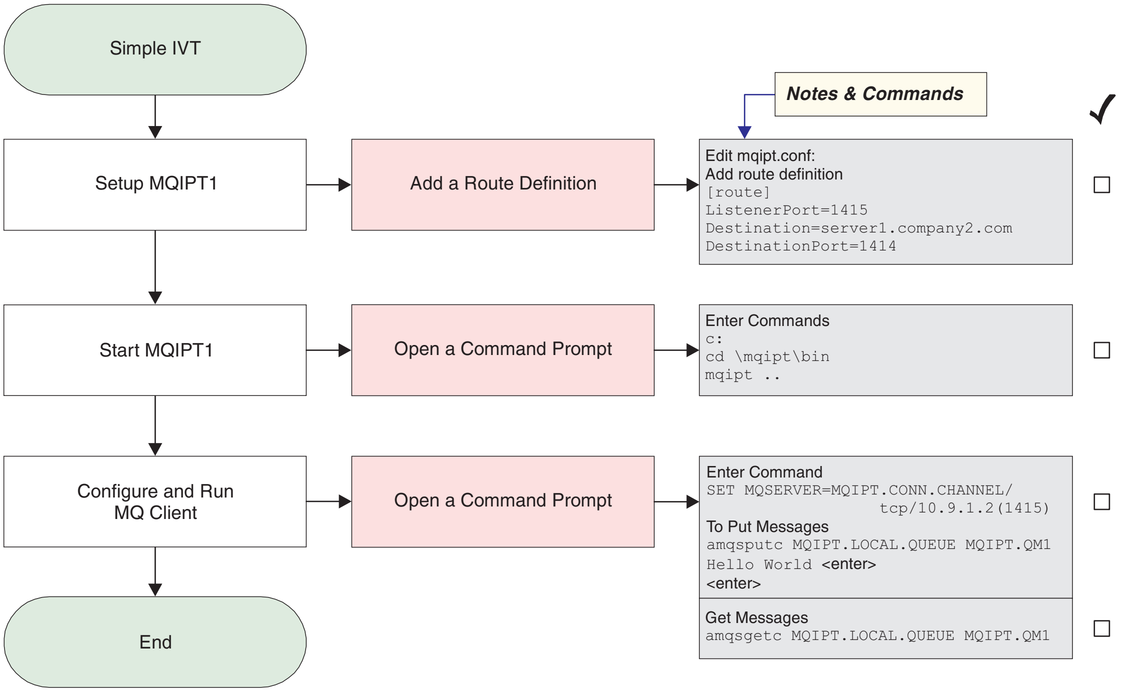


Figure 10. IVT network diagram



Edit mqipt.conf:
Add route definition

```
[route]
ListenerPort=1415
Destination=server1.company2.com
DestinationPort=1414
```

Enter Commands

```
c:
cd \mqipt\bin
mqipt ..
```

Enter Command

```
SET MQSERVER=MQIPT.CONN.CHANNEL/
                    tcp/10.9.1.2(1415)
```

To Put Messages

```
amqsputc MQIPT.LOCAL.QUEUE MQIPT.QM1
Hello World <enter>
<enter>
```

Get Messages

```
amqsgetc MQIPT.LOCAL.QUEUE MQIPT.QM1
```

Figure 11. IVT configuration

1. Setup MQIPT1

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

   ```
   [route]
   ListenerPort=1415
   Destination=server1.company2.com
   DestinationPort=1414
   ```

2. Start MQIPT1

   Open a command prompt and enter the following:

   ```
   c:
   cd \mqipt\bin
   mqipt ..
   ```

   The following message indicates successful completion:

   ```
   5639-L92 (C) Copyright IBM Corp. 2000, 2003 All Rights Reserved
   MQCPI001 WebSphere MQ internet pass-thru Version 1.3.0 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
   ```

3. At a command prompt on the WebSphere MQ client machine, enter the following:

   ```
   SET MQSERVER=MQIPT.CONN.CHANNEL/tcp/10.9.1.2(1415)
   ```

4. Put a message using:

   ```
   amqsputc MQIPT.LOCAL.QUEUE MQIPT1.QM1
   Hello world <enter>
    <enter>
   ```

5. Get the message using:

   ```
   amqsgetc MQIPT.LOCAL.QUEUE MQIPT1.QM1
   ```

You will see "Hello world".

SSL server authentication

In this example you will test an SSL connection using the sample test certificate (`sslsample.pfx` key ring file) by connecting a WebSphere MQ client to a WebSphere MQ server through two MQIPTs. During the SSL handshake, the server will send its test certificate to the client. The client will use its copy of the certificate (with the trust-as-peer flag) to authenticate the server. A default cipher suite, SSL_RSA_WITH_RC4_128_MD5 will be used. (Based on `mqipt.conf` created from "Installation Verification Test" on page 94). For details on how to create a test certificate to use in this example, see "Creating SSL test certificates" on page 112.

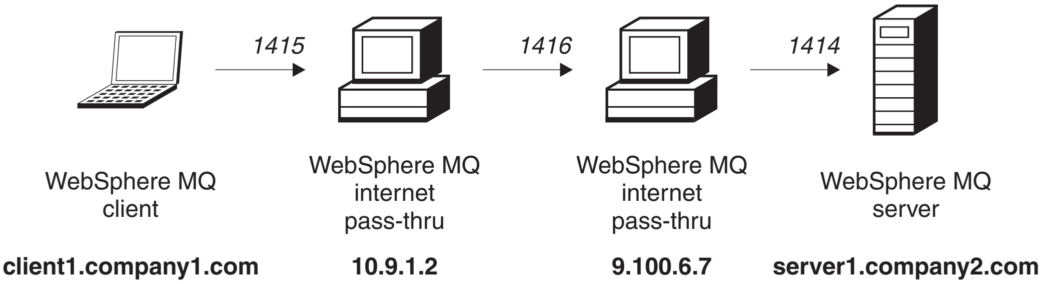
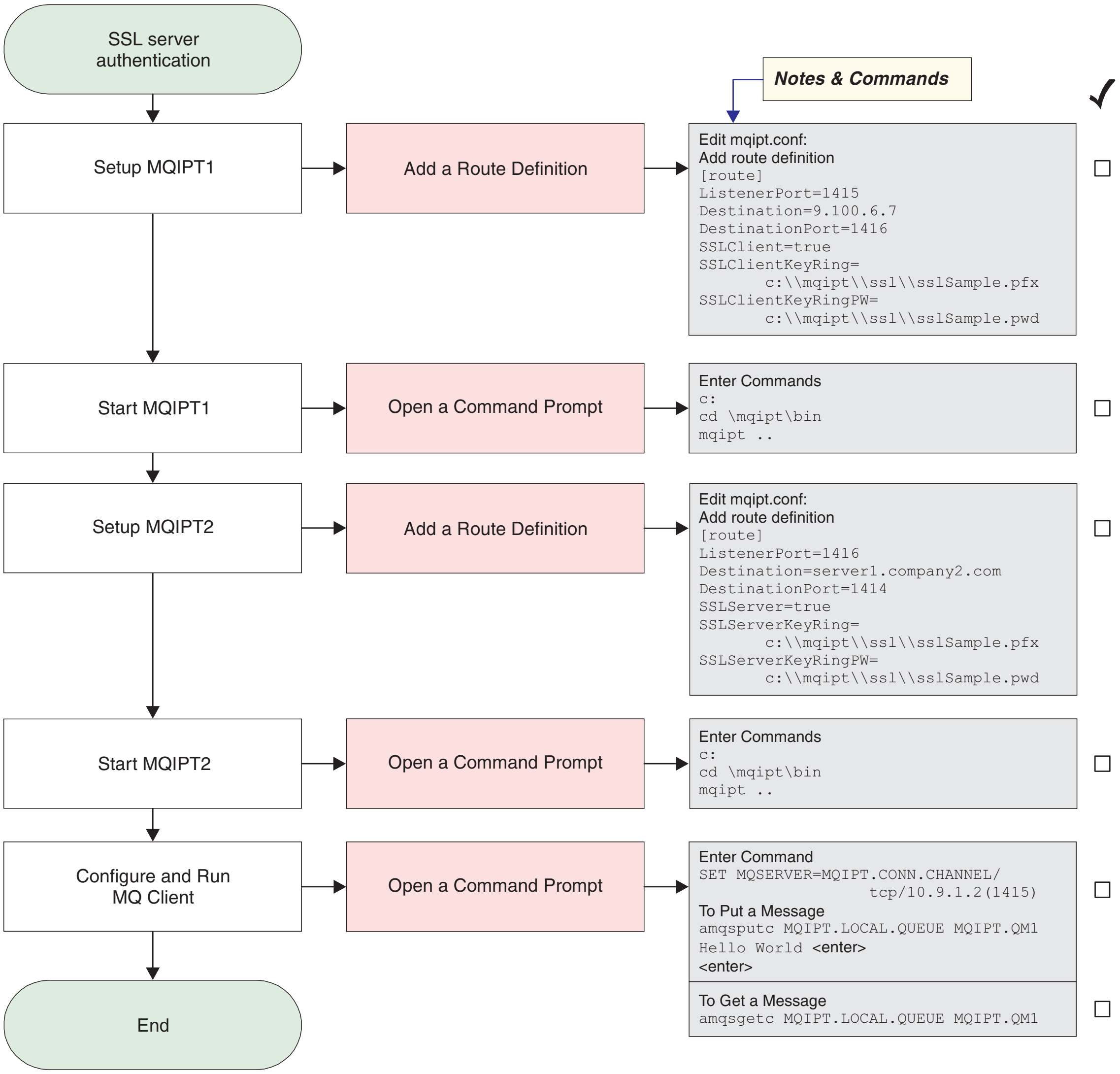


Figure 12. SSL server network diagram



Edit mqipt.conf:
Add 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
```

Enter Commands

```
c:
cd \mqipt\bin
mqipt ..
```

Edit mqipt.conf:
Add 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
```

Enter Commands

```
c:
cd \mqipt\bin
mqipt ..
```

Enter Command

```
SET MQSERVER=MQIPT.CONN.CHANNEL/
                 tcp/10.9.1.2(1415)
```

To Put a Message

```
amqsputc MQIPT.LOCAL.QUEUE MQIPT.QM1
Hello World <enter>
<enter>
```

To Get a Message

```
amqsgetc MQIPT.LOCAL.QUEUE MQIPT.QM1
```

Figure 13. SSL server authentication

1. Setup MQIPT1

   Edit mqipt.conf and add a route definition:

   ```
   [route]
   ListenerPort=1415
   Destination=9.100.6.7
   DestinationPort=1416
   SSLClient=true
   SSLClientKeyRing=C:\\mqipt\\sslSample.pfx
   SSLClientKeyRingPW=C:\\mqipt\\sslSample.pwd
   ```

2. Start MQIPT1

   Open a command prompt and enter the following:

   ```
   c:
   cd \mqipt\bin
   mqipt ..
   ```

   The following message indicates successful completion:

   ```
   5639-L92 (C) Copyright IBM Corp. 2000, 2003 All Rights Reserved
   MQCPI001 WebSphere MQ internet pass-thru Version 1.3.0 starting
   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 ....9.100.6.7(1416)
MQCPI035 ....using MQ protocols
MQCPI036 ....SSL Client side enabled with properties :
MQCPI031 ......cipher suites <null>
MQCPI032 ......keyring file c:\mqipt\sslSample.pfx
MQCPI047 ......CA keyring file <null>
MQCPI038 ......distinguished name(s) CN=* O=* OU=* L=* ST=* C=*
MQCPI078 Route 1415 ready for connection requests
```

3. Setup MQIPT2

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

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

4. Start MQIPT2

   Open a command prompt and enter the following:

   ```
   c:
   cd \mqipt\bin
   mqipt
   ```

   The following message indicates successful completion:

   ```
   5639-L92 (C) Copyright IBM Corp. 2000, 2003 All Rights Reserved
   MQCPI001 WebSphere MQ internet pass-thru Version 1.3.0 starting
   MQCPI011 The path c:\mqipt\logs will be used to store the log files
   MQCPI006 Route 1416 has started and will forward messages to :
   MQCPI034 ....server1.company2.com(1414)
   MQCPI035 ....using MQ protocols
   MQCPI037 ....SSL Server side enabled with properties :
   MQCPI031 ......cipher suites <null>
   MQCPI032 ......keyring file c:\mqipt\sslSample.pfx
   MQCPI047 ......CA keyring file <null>
   MQCPI038 ......distinguished name(s) CN=* O=* OU=* L=* ST=* C=*
   MQCPI033 ......client authentication set to false
   MQCPI078 Route 1416 ready for connection requests
   ```

5. At a command prompt on the WebSphere MQ client machine, enter the following:

   ```
   SET MQSERVER=MQIPT.CONN.CHANNEL/tcp/10.9.1.2(1415)
   ```

6. Put a message using:

   ```
   amqsputc MQIPT.LOCAL.QUEUE MQIPT1.QM1
   Hello world <enter>
    <enter>
   ```

7. Get the message using:

   ```
   amqsgetc MQIPT.LOCAL.QUEUE MQIPT1.QM1
   ```

You will see ″Hello world″.

SSL client authentication

In this example you will test an SSL connection with the sample test certificate. This will perform server and client authentication. During the SSL handshake, the server will send its test certificate to the client. The client will use 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 will use its copy of the certificate, with the trust-as-peer flag, to authenticate the client. A default cipher suite, SSL_RSA_WITH_RC4_128_MD5 will be used. (Based on `mqipt.conf` created from "Installation Verification Test" on page 94).

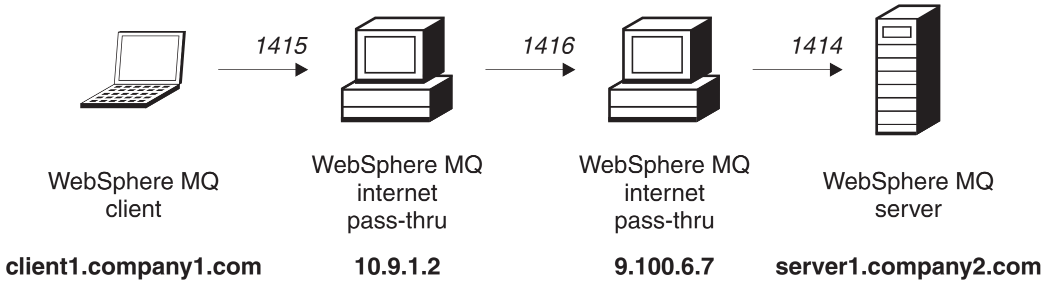
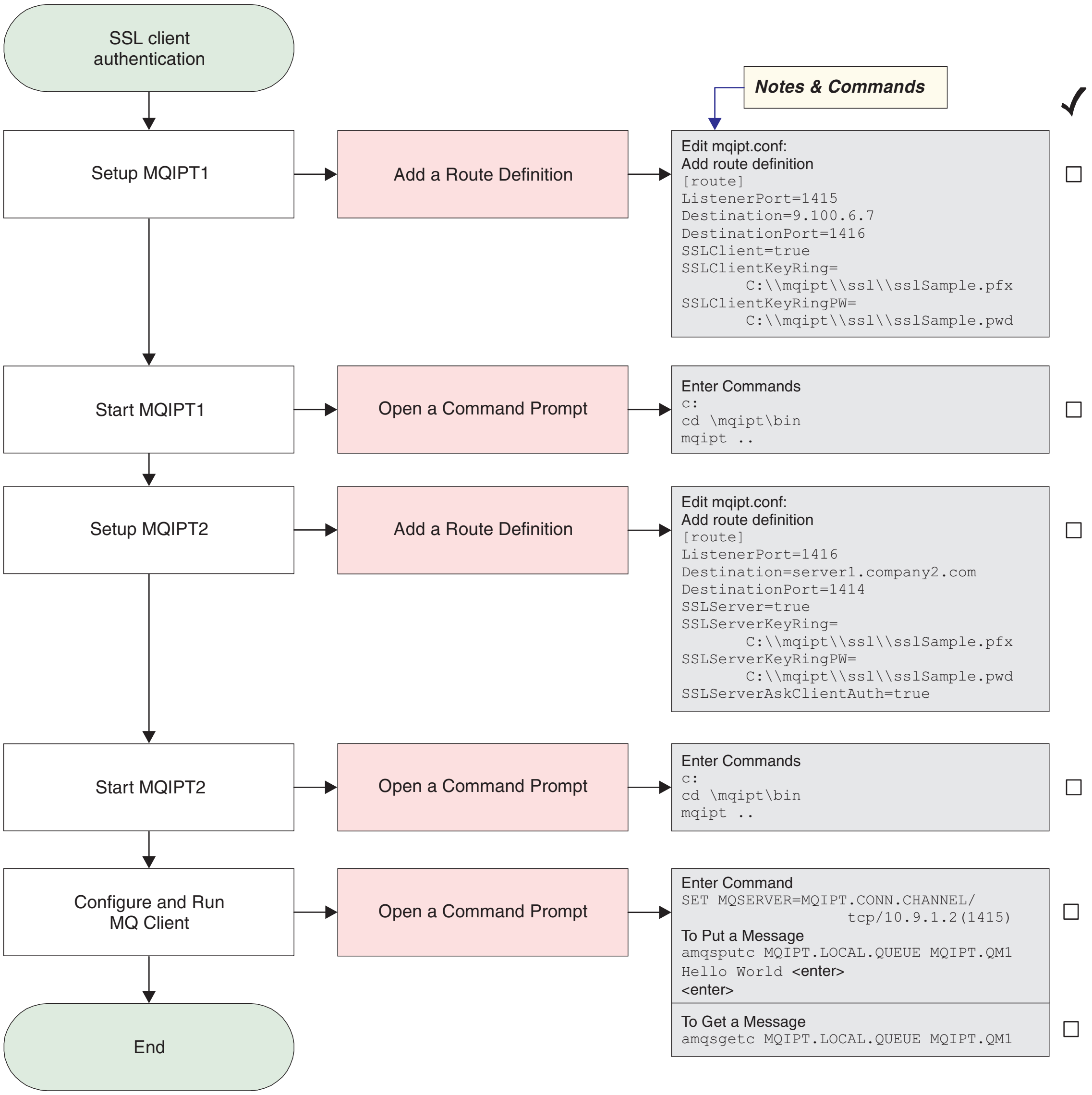


Figure 14. SSL client network diagram



Edit mqipt.conf:
Add 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
```

Enter Commands

```
c:
cd \mqipt\bin
mqipt ..
```

Edit mqipt.conf:
Add 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
SSLServerAskClientAuth=true
```

Enter Commands

```
c:
cd \mqipt\bin
mqipt ..
```

Enter Command

```
SET MQSERVER=MQIPT.CONN.CHANNEL/
                 tcp/10.9.1.2(1415)
```

To Put a Message

```
amqsputc MQIPT.LOCAL.QUEUE MQIPT.QM1
Hello World <enter>
<enter>
```

To Get a Message

```
amqsgetc MQIPT.LOCAL.QUEUE MQIPT.QM1
```

Figure 15. SSL client authentication

1. Setup MQIPT1

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

```
[route]
ListenerPort=1415
Destination=9.100.6.7
DestinationPort=1416
SSLClient=true
SSLClientKeyRing=C:\\mqipt\\sslSample.pfx
SSLClientKeyRingPW=C:\\mqipt\\sslSample.pwd
```

2. Start MQIPT1

   Open a command prompt and enter the following:

   ```
   c:
   cd \mqipt\bin
   mqipt ..
   ```

   The following message indicates successful completion:

   ```
   5639-L92 (C) Copyright IBM Corp. 2000, 2003 All Rights Reserved
   MQCPI001 WebSphere MQ internet pass-thru Version 1.3.0 starting
   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 ....9.100.6.7(1416)
   MQCPI035 ....using MQ protocols
   MQCPI036 ....SSL Client side enabled with properties :
   MQCPI031 ......cipher suites <null>
   MQCPI032 ......keyring file c:\mqipt\sslSample.pfx
   MQCPI047 ......CA keyring file <null>
   MQCPI038 ......distinguished name(s) CN=* O=* OU=* L=* ST=* C=*
   MQCPI078 Route 1415 ready for connection requests
   ```

3. Setup MQIPT2

   Edit mqipt.conf and add a route definition:

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

4. Start MQIPT2

   Open a command prompt and enter the following:

   ```
   c:
   cd \mqipt\bin
   mqipt
   ```

   The following message indicates successful completion:

   ```
   5639-L92 (C) Copyright IBM Corp. 2000, 2003 All Rights Reserved
   MQCPI001 WebSphere MQ internet pass-thru Version 1.3.0 starting
   MQCPI011 The path c:\mqipt\logs will be used to store the log files
   MQCPI006 Route 1416 has started and will forward messages to :
   MQCPI034 ....server1.company2.com(1414)
   MQCPI035 ....using MQ protocols
   MQCPI037 ....SSL Server side enabled with properties :
   MQCPI031 ......cipher suites <null>
   MQCPI032 ......keyring file c:\mqipt\sslSample.pfx
   MQCPI047 ......CA keyring file <null>
   MQCPI038 ......distinguished name(s) CN=* O=* OU=* L=* ST=* C=*
   MQCPI033 ......client authentication set to true
   MQCPI078 Route 1416 ready for connection requests
   ```

5. At a command prompt on the WebSphere MQ client machine, enter the following:

   ```
   SET MQSERVER=MQIPT.CONN.CHANNEL/tcp/10.9.1.2(1415)
   ```

6. Put a message using:

```
amqsputc MQIPT.LOCAL.QUEUE MQIPT1.QM1
Hello world <enter>
 <enter>
```

7. Get the message using:

   ```
   amqsgetc MQIPT.LOCAL.QUEUE MQIPT1.QM1
   ```

You will see ″Hello world″.

HTTP proxy configuration

In this example you will test the connection using an HTTP proxy (IBM Caching Proxy). CP must be at level 3.6 or greater, you must also check the following:

- `ProxyPersistence` must be `on`, this allows for persistent connections
- `MaxPersistRequest` 5000, this is the number of requests allowed on a single connection before the connection is broken
- `PersistTimeout` 12hrs, this is the time allowed for the connection to exist

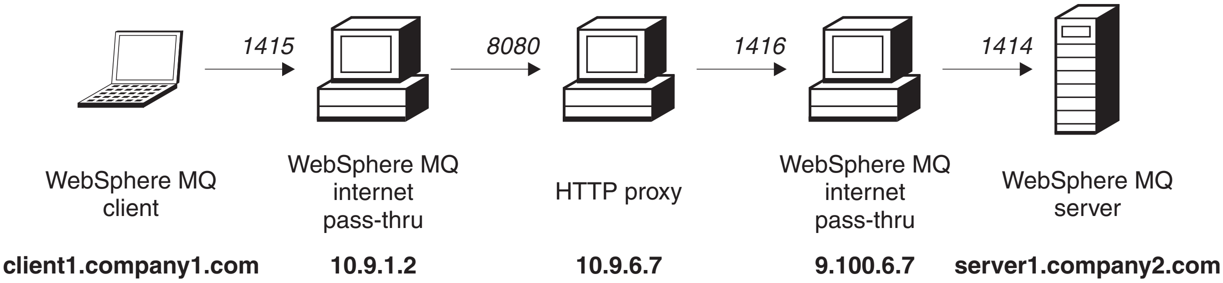
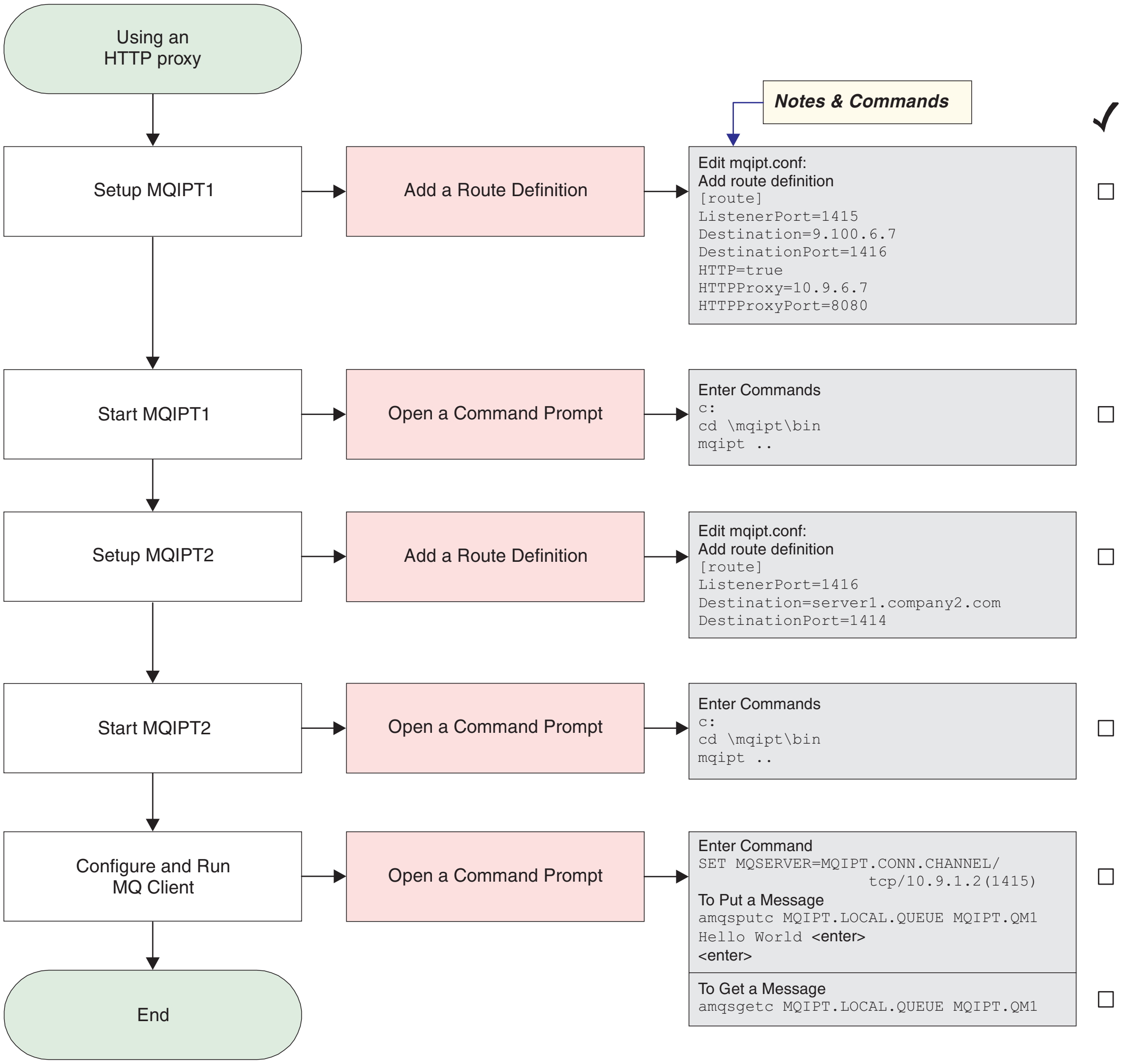


Figure 16. HTTP proxy network diagram



Edit mqipt.conf:
Add route definition

```
[route]
ListenerPort=1415
Destination=9.100.6.7
DestinationPort=1416
HTTP=true
HTTPProxy=10.9.6.7
HTTPProxyPort=8080
```

□

Enter Commands

```
c:
cd \mqipt\bin
mqipt ..
```

□

Edit mqipt.conf:
Add route definition

```
[route]
ListenerPort=1416
Destination=server1.company2.com
DestinationPort=1414
```

□

Enter Commands

```
c:
cd \mqipt\bin
mqipt ..
```

□

Enter Command

```
SET MQSERVER=MQIPT.CONN.CHANNEL/
                  tcp/10.9.1.2(1415)
```

To Put a Message

```
amqsputc MQIPT.LOCAL.QUEUE MQIPT.QM1
Hello World <enter>
<enter>
```

□

To Get a Message

```
amqsgetc MQIPT.LOCAL.QUEUE MQIPT.QM1
```

□

Figure 17. HTTP proxy configuration

1. Setup MQIPT1

   Edit mqipt.conf and add a route definition:

   ```
   [route]
   ListenerPort=1415
   Destination=9.100.6.7
   DestinationPort=1416
   HTTP=true
   HTTPProxy=true
   HTTPProxyPort=8080
   ```

2. Start MQIPT1

   Open a command prompt and enter the following:

   ```
   c:
   cd \mqipt\bin
   mqipt ..
   ```

   The following message indicates successful completion:

   ```
   5639-L92 (C) Copyright IBM Corp. 2000, 2003 All Rights Reserved
   MQCPI001 WebSphere MQ internet pass-thru Version 1.3.0 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 ....9.100.6.7(1416)
MQCPI035 ....using HTTP
MQCPI024 ....and HTTP proxy at 10.9.6.7(1080)
MQCPI078 Route 1415 ready for connection requests
```

3. Setup MQIPT2

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

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

4. Start MQIPT2

   Open a command prompt and enter the following:

   ```
   c:
   cd \mqipt\bin
   mqipt
   ```

   The following message indicates successful completion:

   ```
   5639-L92 (C) Copyright IBM Corp. 2000, 2003 All Rights Reserved
   MQCPI001 WebSphere MQ internet pass-thru Version 1.3.0 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 1416 has started and will forward messages to :
   MQCPI034 ....server1.company2.com(1414)
   MQCPI035 ....using MQ protocols
   MQCPI078 Route 1416 ready for connection requests
   ```

5. At a command prompt on the WebSphere MQ client machine, enter the following:

   ```
   SET MQSERVER=MQIPT.CONN.CHANNEL/tcp/10.9.1.2(1415)
   ```

6. Put a message using:

   ```
   amqsputc MQIPT.LOCAL.QUEUE MQIPT1.QM1
   Hello world <enter>
    <enter>
   ```

7. Get the message using:

   ```
   amqsgetc MQIPT.LOCAL.QUEUE MQIPT1.QM1
   ```

You will see ″Hello world″.

Configuring access control

In this example you will set up your MQIPT to only accept connections from specific clients by adding security checks on the MQIPT listener port, using the Java Security Manager.

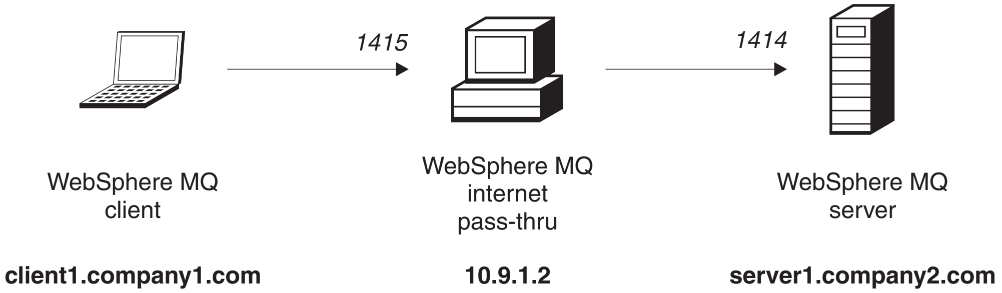
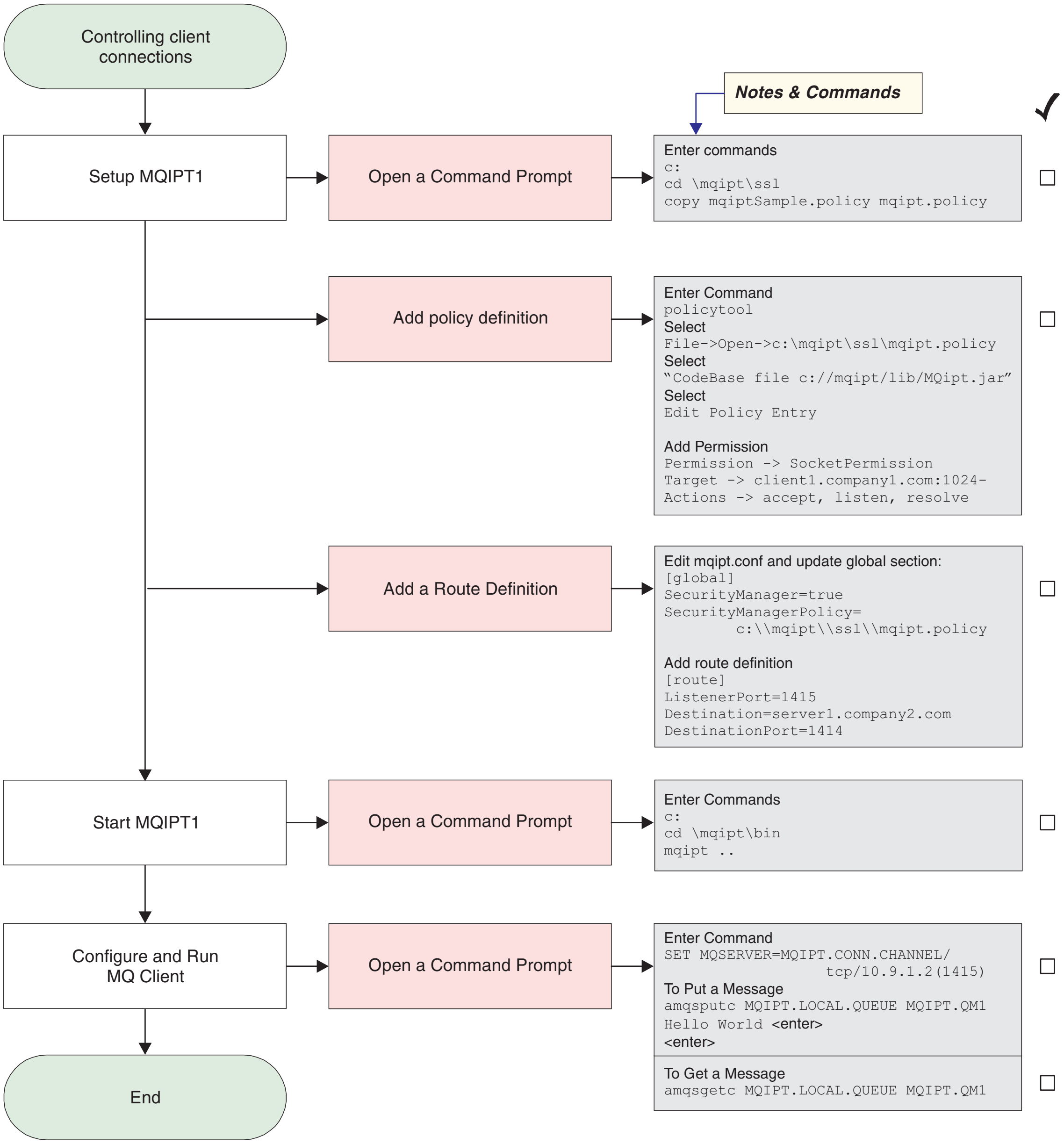


Figure 18. Access control network diagram



Enter commands

```
c:
cd \mqipt\ssl
copy mqiptSample.policy mqipt.policy
```

□

Enter Command

```
policytool
```

Select

```
File->Open->c:\mqipt\ssl\mqipt.policy
```

Select

```
“CodeBase file c://mqipt/lib/MQipt.jar”
```

Select

```
Edit Policy Entry
```

Add Permission

```
Permission -> SocketPermission
Target -> client1.company1.com:1024-
Actions -> accept, listen, resolve
```

□

Edit mqipt.conf and update global section:

```
[global]
SecurityManager=true
SecurityManagerPolicy=
        c:\\mqipt\\ssl\\mqipt.policy
```

Add route definition

```
[route]
ListenerPort=1415
Destination=server1.company2.com
DestinationPort=1414
```

□

Enter Commands

```
c:
cd \mqipt\bin
mqipt ..
```

□

Enter Command

```
SET MQSERVER=MQIPT.CONN.CHANNEL/
                tcp/10.9.1.2(1415)
```

To Put a Message

```
amqsputc MQIPT.LOCAL.QUEUE MQIPT.QM1
Hello World <enter>
<enter>
```

□

To Get a Message

```
amqsgetc MQIPT.LOCAL.QUEUE MQIPT.QM1
```

□

Figure 19. Access control configuration

1. Setup MQIPT1
   a. Open a command prompt and enter the following:
      ```
      c:
      cd \mqipt\ssl
      copy c:\mqipt\ssl\mqiptSample.policy to mqipt.policy
      ```
   b. Add a policy definition using the following command:
      `policytool`
      1) Select File -> Open -> `c:\mqipt\ssl\mqipt.policy`
      2) Select:
         `file://C:/Program Files/IBM/WebSphere MQ internet pass-thru/lib/MQipt.jar`
      3) Change CodeBase from:
         `file://C:/Program Files/IBM/WebSphere MQ internet pass-thru/lib/MQipt.jar`

        to:

        ```
        file://C:/mqipt/lib/MQipt.jar
        ```

    4) Change all permissions from:

        ```
        C:\\Program Files\\IBM\\WebSphere MQ internet pass-thru
        ```

        to:

        ```
        C:\\mqipt
        ```

    5) Add SocketPermission:

        ```
        Permission=SocketPermission
        Target=client1.company1.com:1024-
        Actions=accept, listen, resolve
        ```

   c. Edit mqipt.conf and add:

    1) Two properties to the global section:

        ```
        [global]
        SecurityManager=true
        SecurityManagerPolicy=c:\\mqipt\\ssl\\mqipt.policy
        ```

    2) A route definition:

        ```
        [route]
        ListenerPort=1415
        Destination=server1.company2.com
        DestinationPort=1414
        ```

2. Start MQIPT1

   Open a command prompt and enter the following:

   ```
   c:
   cd \mqipt\bin
   mqipt ..
   ```

   The following message indicates successful completion:

   ```
   5639-L92 (C) Copyright IBM Corp. 2000, 2003 All Rights Reserved
   MQCPI001 WebSphere MQ internet pass-thru Version 1.3.0 starting
   MQCPI004 Reading configuration information from C:\mqipt\mqipt.conf
   MQCPI055 Setting the java.security.policy to c:\mqipt\mqipt.policy
   MQCPI053 Starting the Java Security Manager
   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
   ```

3. At a command prompt on the WebSphere MQ client machine, enter the following:

   ```
   SET MQSERVER=MQIPT.CONN.CHANNEL/tcp/10.9.1.2(1415)
   ```

4. Put a message using:

   ```
   amqsputc MQIPT.LOCAL.QUEUE MQIPT1.QM1
   Hello world <enter>
    <enter>
   ```

5. Get the message using:

   ```
   amqsgetc MQIPT.LOCAL.QUEUE MQIPT1.QM1
   ```

You will see ″Hello world″.

Configuring Quality of Service (QoS)

For this example, we assume that TQoS has already been installed on the same machine as MQIPT.

In this example you will apply a Quality of Service (QoS) to all channels on an MQIPT route. This can only be implemented when running MQIPT on the Linux platform. This sample will set a priority of ″average″ for all data sent from MQIPT to the WebSphere MQ client and a priority of ″good″ for all data sent to the WebSphere MQ server. Using the sample pagent policies listed below, the following priorities can be applied to `QosToCaller` and `QosToDest`:

- 1 - average
- 2 - good
- 3 - very good

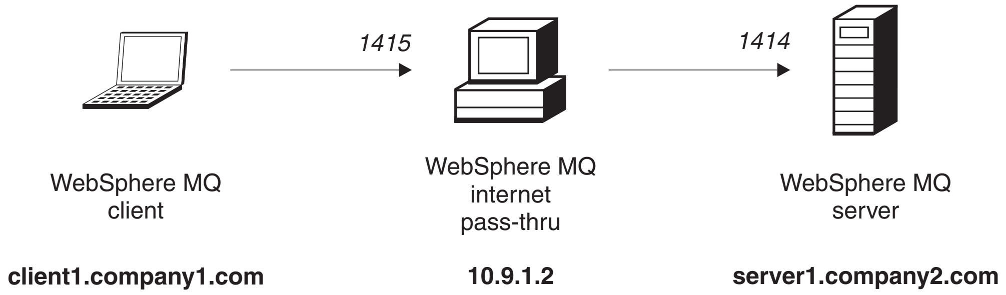
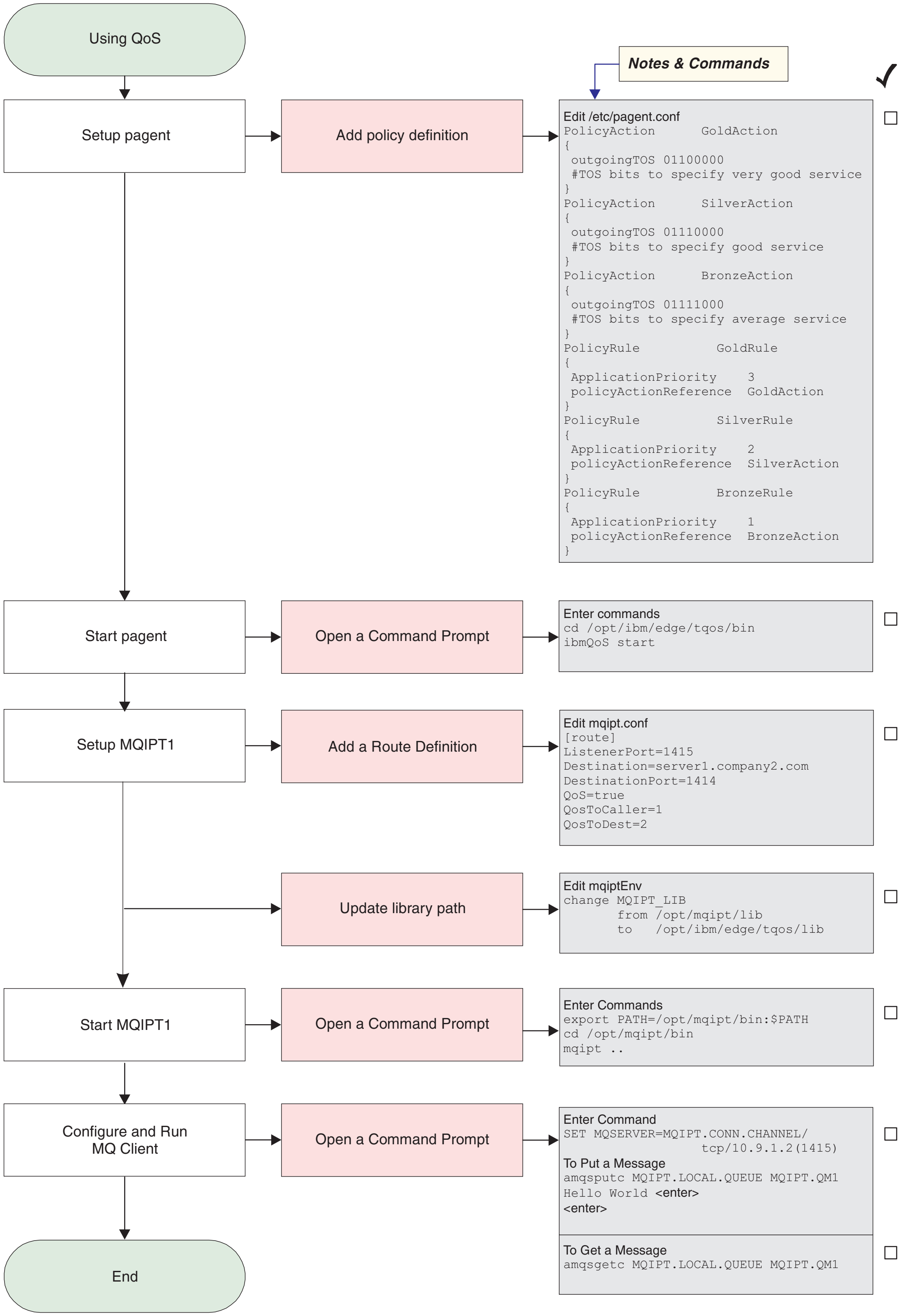


Figure 20. QoS network diagram



Edit /etc/pagent.conf □

```
PolicyAction      GoldAction
{
 outgoingTOS 01100000
 #TOS bits to specify very good service
}
PolicyAction      SilverAction
{
 outgoingTOS 01110000
 #TOS bits to specify good service
}
PolicyAction      BronzeAction
{
 outgoingTOS 01111000
 #TOS bits to specify average service
}
PolicyRule          GoldRule
{
 ApplicationPriority    3
 policyActionReference  GoldAction
}
PolicyRule          SilverRule
{
 ApplicationPriority    2
 policyActionReference  SilverAction
}
PolicyRule          BronzeRule
{
 ApplicationPriority    1
 policyActionReference  BronzeAction
}
```

Enter commands □

```
cd /opt/ibm/edge/tqos/bin
ibmQoS start
```

Edit mqipt.conf □

```
[route]
ListenerPort=1415
Destination=server1.company2.com
DestinationPort=1414
QoS=true
QosToCaller=1
QosToDest=2
```

Edit mqiptEnv □

```
change MQIPT_LIB
       from /opt/mqipt/lib
       to   /opt/ibm/edge/tqos/lib
```

Enter Commands □

```
export PATH=/opt/mqipt/bin:$PATH
cd /opt/mqipt/bin
mqipt ..
```

Enter Command □

```
SET MQSERVER=MQIPT.CONN.CHANNEL/
                  tcp/10.9.1.2(1415)
```

To Put a Message

```
amqsputc MQIPT.LOCAL.QUEUE MQIPT.QM1
Hello World <enter>
<enter>
```

To Get a Message □

```
amqsgetc MQIPT.LOCAL.QUEUE MQIPT.QM1
```

Figure 21. QoS configuration

1. Set up pagent

   Edit `/etc/pagent.conf` and add the following:

```
PolicyAction      GoldAction
{
 outgoingTOS 01100000
 #TOS bits to specify very good service
}
PolicyAction      SilverAction
{
 outgoingTOS 01110000
 #TOS bits to specify good service
}
PolicyAction      BronzeAction
{
 outgoingTOS 01111000
 #TOS bits to specify average service
}
PolicyRule          GoldRule
{
 ApplicationPriority    3
 policyActionReference  GoldAction
}
PolicyRule          SilverRule
{
 ApplicationPriority    2
 policyActionReference  SilverAction
}
PolicyRule          BronzeRule
{
 ApplicationPriority    1
 policyActionReference  BronzeAction
}
```

To turn on the performance data collection for the rules defined above, use statement PolicyPerformanceCollection and enable it. Please refer to Pagent.conf for a description and format of this statement.

2. Start pagent

   Open a command prompt and enter the following:

   ```
   cd /opt/ibm/edge/tqos/bin
   ibmQoS start
   ```

3. Setup MQIPT1

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

   ```
   [route]
   ListenerPort=1415
   Destination=server1.company2.com
   DestinationPort=1414
   QoS=true
   QosToCaller=1
   QosToDest=2
   ```

4. Update the library path

   Edit `mqiptEnv` (found in `/opt/mqipt/bin`) and change MQIPT_LIB from:

   ```
   /opt/mqipt/lib
   ```

   to:

   ```
   /opt/ibm/edge/tqos/lib
   ```

5. Start MQIPT1

   Open a command prompt and enter the following:

   ```
   export PATH=/opt/mqipt/bin:$PATH
   cd /opt/mqipt/bin
   mqipt ..
   ```

   The following message indicates successful completion:

```
5639-L92 (C) Copyright IBM Corp. 2000, 2003 All Rights Reserved
MQCPI001 Websphere MQ internet pass-thru Version 1.3.0 starting
MQCPI004 Reading configuration information from /opt/mqipt/mqipt.conf
MQCPI011 The path /opt/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
MQCPI049 ....QoS priority to dest = 2, to caller = 1
MQCPI078 Route 1415 ready for connection requests
```

6. At a command prompt on the WebSphere MQ client machine, enter the following:

   ```
   SET MQSERVER=MQIPT.CONN.CHANNEL/tcp/10.9.1.2(1415)
   ```

7. Put a message using:

   ```
   amqsputc MQIPT.LOCAL.QUEUE MQIPT1.QM1
   Hello world <enter>
    <enter>
   ```

8. Get the message using:

   ```
   amqsgetc MQIPT.LOCAL.QUEUE MQIPT1.QM1
   ```

You will see ″Hello world″.

Configuring SOCKS proxy

In this example you can make MQIPT act as a SOCKS proxy. The WebSphere MQ client must be socksified before running this sample and the SOCKS configuration must point to MQIPT as the SOCKS proxy. The definitions of the MQIPT Destination and DestinationPort properties can be anything, as the true destination is obtained from the WebSphere MQ client during the socks handshaking process.

Before starting, you must either socksify the whole machine or just the WebSphere MQ client application (amqsputc/amqsgetc). You must also configure the SOCKS client to:

- Point to MQIPT as the Socks proxy
- Enable Socks V5 support
- Disable user authentication
- Only make connections to the MQIPT network address

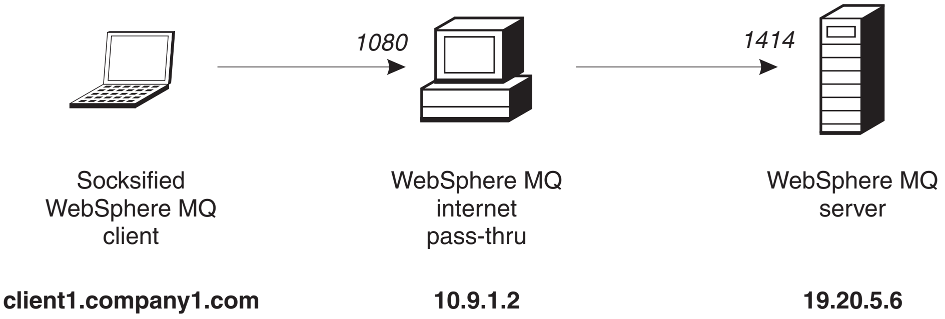
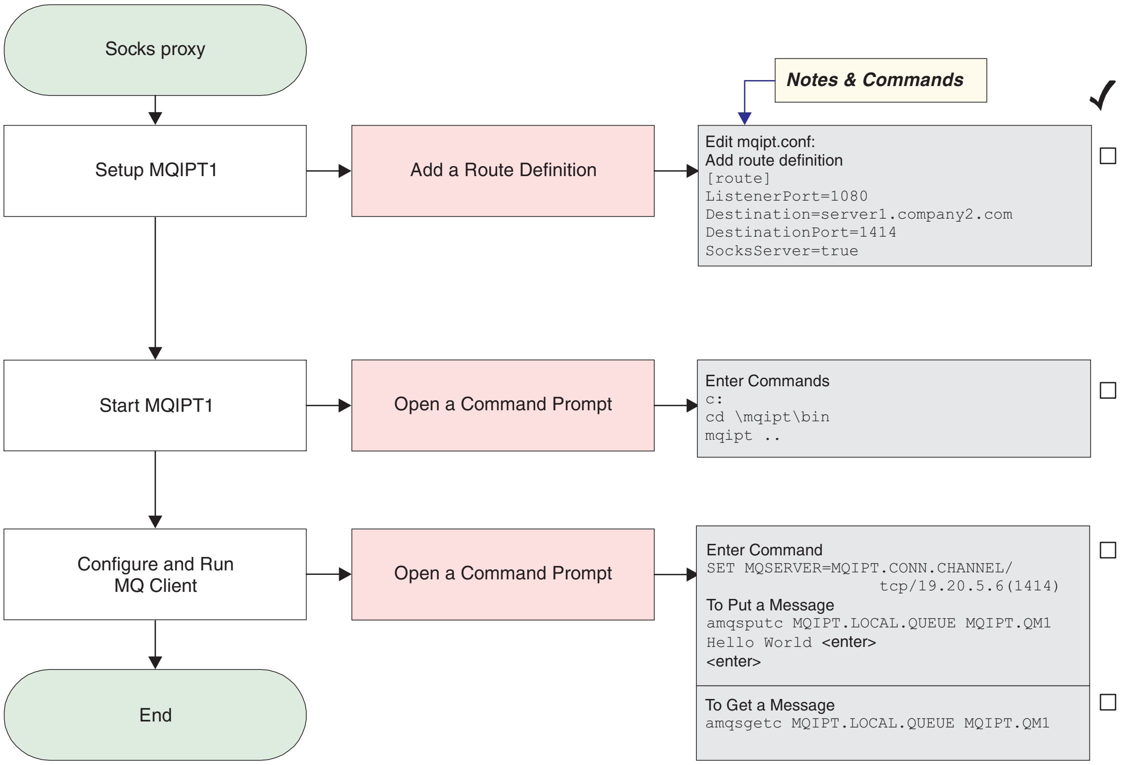


Figure 22. SOCKS proxy network diagram



Edit mqipt.conf:
Add route definition

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

Enter Commands

```
c:
cd \mqipt\bin
mqipt ..
```

Enter Command

```
SET MQSERVER=MQIPT.CONN.CHANNEL/
                     tcp/19.20.5.6(1414)
```

To Put a Message

```
amqsputc MQIPT.LOCAL.QUEUE MQIPT.QM1
Hello World <enter>
<enter>
```

To Get a Message

```
amqsgetc MQIPT.LOCAL.QUEUE MQIPT.QM1
```

Figure 23. SOCKS proxy configuration

1. Setup MQIPT1

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

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

2. Start MQIPT1

   Open a command prompt and enter the following:

   ```
   c:
   cd \mqipt\bin
   mqipt ..
   ```

   The following message indicates successful completion:

   ```
   5639-L92 (C) Copyright IBM Corp. 2000, 2003 All Rights Reserved
   MQCPI001 WebSphere MQ internet pass-thru Version 1.3.0 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 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
   ```

3. At a command prompt on the WebSphere MQ client machine, enter the following:

   ```
   SET MQSERVER=MQIPT.CONN.CHANNEL/tcp/19.20.5.6(1414)
   ```

4. Put a message using:

   ```
   amqsputc MQIPT.LOCAL.QUEUE MQIPT1.QM1
   Hello world <enter>
    <enter>
   ```

5. Get the message using:

```
amqsgetc MQIPT.LOCAL.QUEUE MQIPT1.QM1
```

You will see "Hello world".

Configuring SOCKS client

In this example you will run MQIPT as though it was socksified, using an existing SOCKS proxy. This is similar to “Configuring SOCKS proxy” on page 109, except MQIPT makes a socksified connection, instead of the WebSphere MQ client.

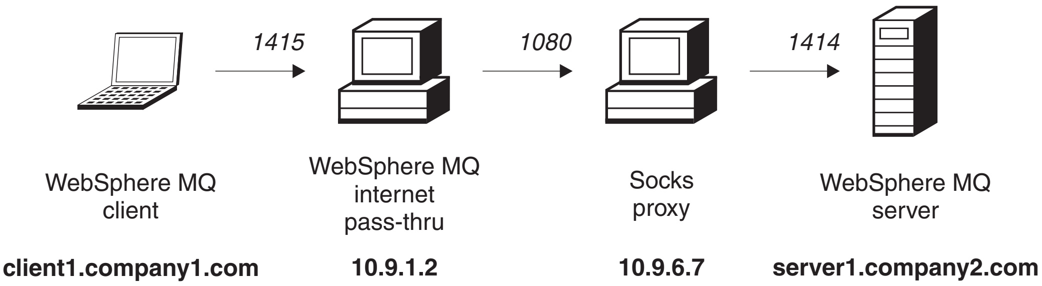
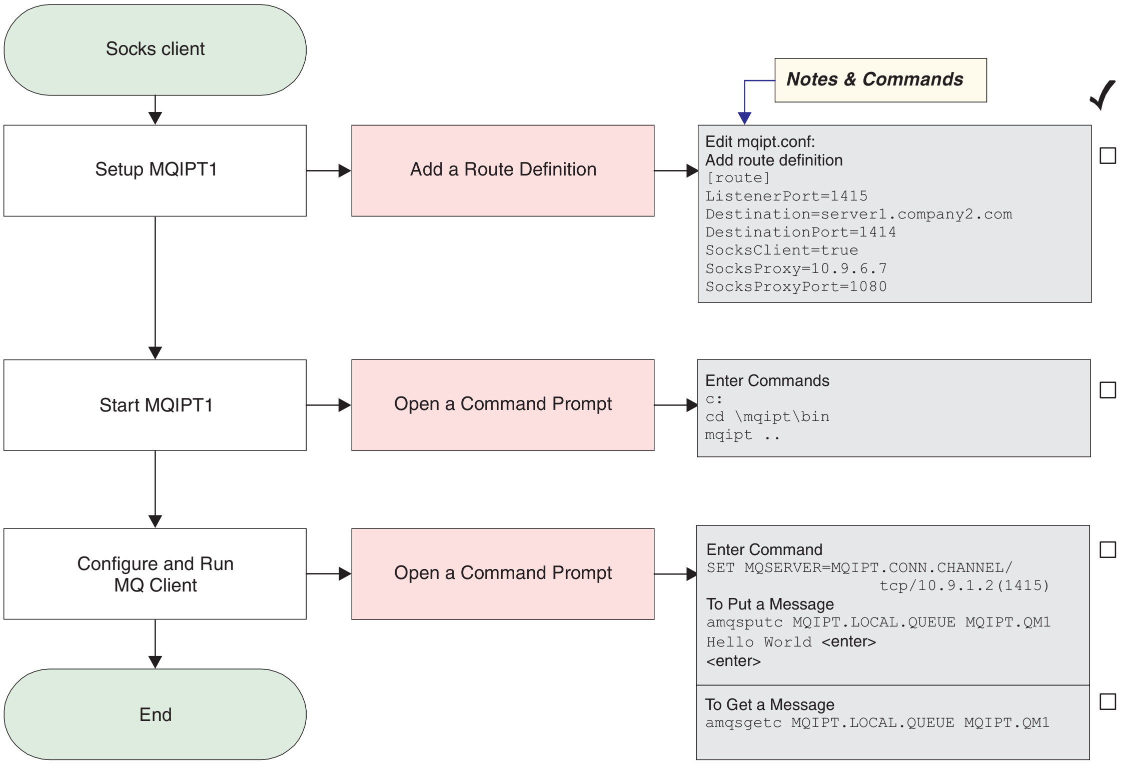


Figure 24. SOCKS client network diagram



Edit mqipt.conf:
Add route definition

```
[route]
ListenerPort=1415
Destination=server1.company2.com
DestinationPort=1414
SocksClient=true
SocksProxy=10.9.6.7
SocksProxyPort=1080
```

Enter Commands

```
c:
cd \mqipt\bin
mqipt ..
```

Enter Command

```
SET MQSERVER=MQIPT.CONN.CHANNEL/
                  tcp/10.9.1.2(1415)
```

To Put a Message

```
amqsputc MQIPT.LOCAL.QUEUE MQIPT.QM1
Hello World <enter>
<enter>
```

To Get a Message

```
amqsgetc MQIPT.LOCAL.QUEUE MQIPT.QM1
```

Figure 25. SOCKS client configuration

1. Setup MQIPT1

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

```
[route]
ListenerPort=1415
Destination=server1.company2.com
```

```
DestinationPort=1414
SocksClient=true
SocksProxy=10.9.6.7
SocksProxyPort=1080
```

2. Start MQIPT1

   Open a command prompt and enter the following:

   ```
   c:
   cd \mqipt\bin
   mqipt ..
   ```

   The following message indicates successful completion:

   ```
   5639-L92 (C) Copyright IBM Corp. 2000, 2003 All Rights Reserved
   MQCPI001 WebSphere MQ internet pass-thru Version 1.3.0 starting
   MQCPI004 Reading configuration information from C:\mqipt\mqipt.conf
   MQCPI022 Password checking has been disabled on the command port
   MQCPI011 The path C:\mqipt\logs will be used to store the log files
   MQCPI006 Route 1415 has started and will forward messages to :
   MQCPI034 ....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 WebSphere MQ client machine, enter the following:

   ```
   SET MQSERVER=MQIPT.CONN.CHANNEL/tcp/10.9.1.2(1415)
   ```

4. Put a message using:

   ```
   amqsputc MQIPT.LOCAL.QUEUE MQIPT1.QM1
   Hello world <enter>
    <enter>
   ```

5. Get the message using:

   ```
   amqsgetc MQIPT.LOCAL.QUEUE MQIPT1.QM1
   ```

You will see ″Hello world″.

Creating SSL test certificates

In this example we will show you how to create a self-signed certificate which can be used for testing MQIPT routes. The certificate will have the trust-as-peer flag turned on.

1. Start KeyMan
2. Select ″Create new...″
3. Select ″PKCS#12 Token″
4. Select ″Action -> Generate Key″

   a new key pair will appear in the list ″RSA / 1024-bit″
5. Select the new key pair
6. Select ″Action -> Create Certificate″
7. Select ″Self-signed Certificate″
8. Enter certificate details.

   You will see a dialog explaining the private certificate will be joined with the key, entering a label is optional
9. Select the new certificate
10. Display certificate details
11. Change certificate properties
12. Turn on trust-as-peer flag

13. Close dialog Select ″File -> Save″
14. Enter passphrase (for example, myPassWord)
15. Enter a file name of new key ring file (for example, c:\mqipt\ssl\testRoute1414.pfx)

    You must keep ″File format as PKCS#12 / PFX″ - **do not check** ″Wrap key ring into a Java class″
16. Create a text file containing the passphrase (myPassWord) you used above.

    For example, c:\mqipt\ssl\testRoute1414.pwd

This key ring file can now be used in the example “SSL server authentication” on page 96.

Configuring the MQIPT Servlet

In addition to “Assumptions” on page 93, this example also makes the following assumptions:

- The Tomcat Application Server has been installed in the following directory:

  ```
  c:\jakarta-tomcat-4.0.1
  ```

  You can download Tomcat from:

  ```
  http://jakarta.apache.org/builds/jakarta-tomcat-4.0/release/v4.0.3/
  ```
- IBM Web Traffic Express has been installed in:

  ```
  c:\wte
  ```

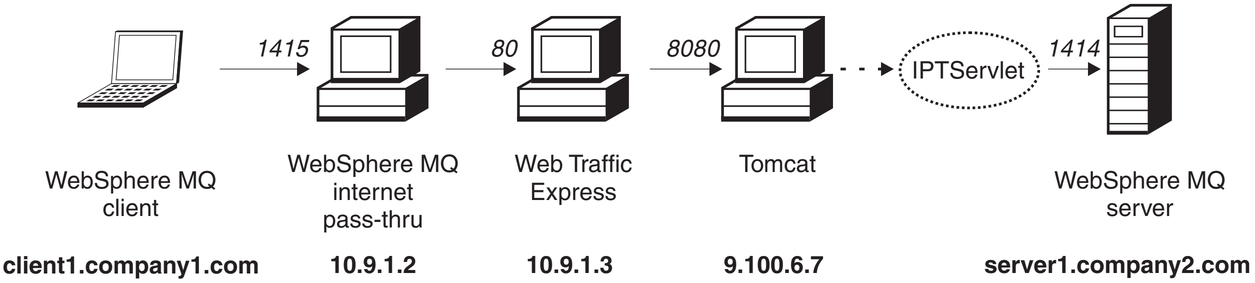
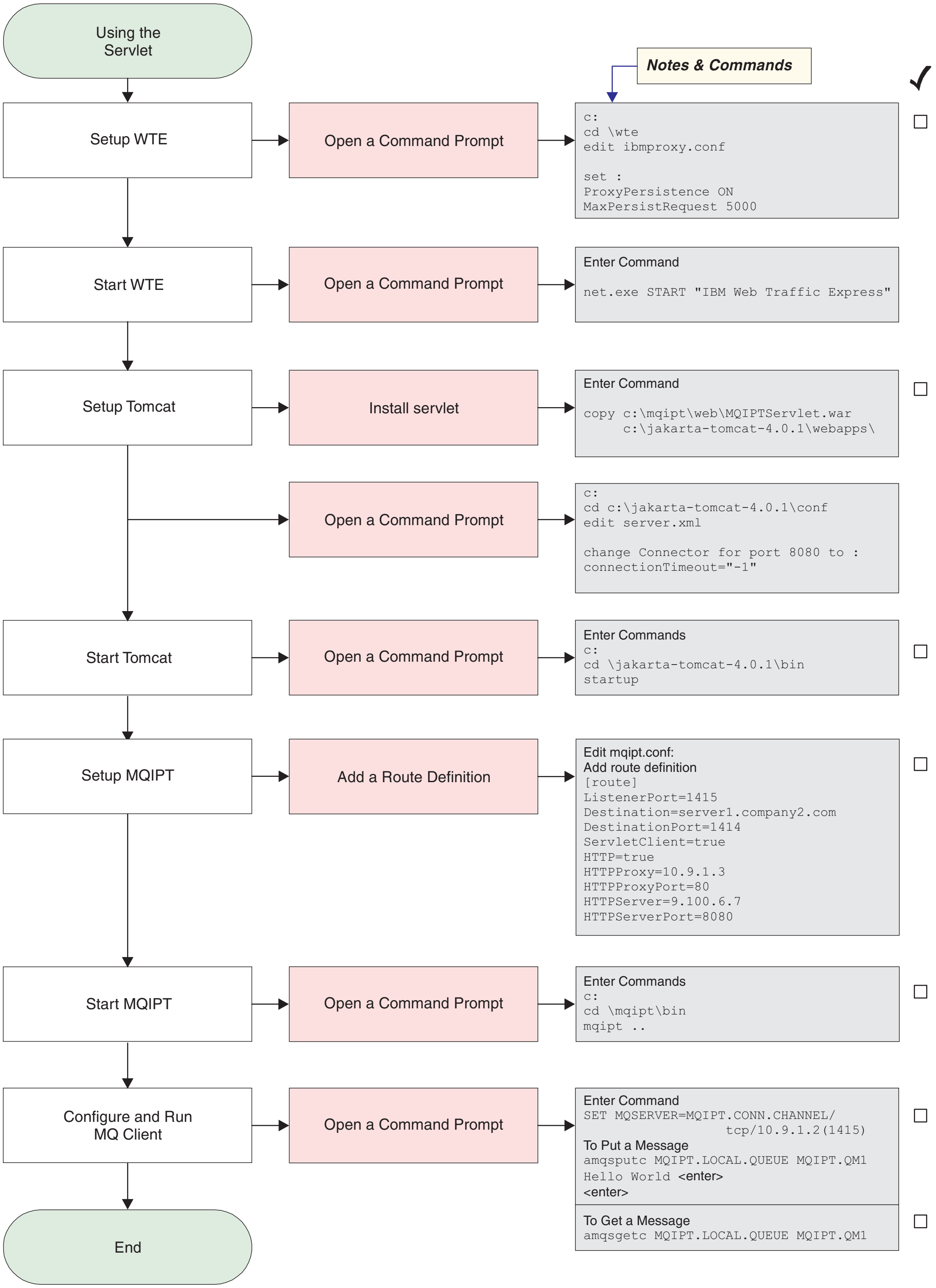


Figure 26. Servlet network diagram



```
c:
cd \wte
edit ibmproxy.conf

set :
ProxyPersistence ON
MaxPersistRequest 5000
```

```
net.exe START "IBM Web Traffic Express"
```

```
copy c:\mqipt\web\MQIPTServlet.war
     c:\jakarta-tomcat-4.0.1\webapps\
```

```
c:
cd c:\jakarta-tomcat-4.0.1\conf
edit server.xml

change Connector for port 8080 to :
connectionTimeout="-1"
```

```
c:
cd \jakarta-tomcat-4.0.1\bin
startup
```

```
[route]
ListenerPort=1415
Destination=server1.company2.com
DestinationPort=1414
ServletClient=true
HTTP=true
HTTPProxy=10.9.1.3
HTTPProxyPort=80
HTTPServer=9.100.6.7
HTTPServerPort=8080
```

```
c:
cd \mqipt\bin
mqipt ..
```

```
SET MQSERVER=MQIPT.CONN.CHANNEL/
                  tcp/10.9.1.2(1415)
```

```
amqsputc MQIPT.LOCAL.QUEUE MQIPT.QM1
Hello World <enter>
<enter>
```

```
amqsgetc MQIPT.LOCAL.QUEUE MQIPT.QM1
```

Figure 27. Servlet configuration

1. Setup Web Traffic Express

   edit `c:\wte\ibmroxy.conf` and set the following properties:

   ```
   ProxyPersistence ON
   MaxPersistRequest  5000
   ```

2. Start Web Traffic Express

   Open a command prompt and enter the following:

   ```
   net.exe Start "IBM Web Traffic Express"
   ```

3. Setup Tomcat

   To install the Servlet, copy:

   ```
   c:\mqipt\web\MQIPTServlet.war
   ```

   to:

   ```
   c:\jakarta-tomcat-4.0.1\webapps
   ```

   Edit `c:\jakarta-tomcat-4.0.1\conf\server.xml`, enable the connector for port 8443 and set the `ConnectionTimeout` property to `-1`.

4. Start Tomcat

   Open a command prompt and enter the following:

   ```
   c:
   cd \jakarta-tomcat-4.0.1\bin
   startup
   ```

5. Setup MQIPT1

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

   ```
   [route]
   ListenerPort=1415
   Destination=server1.company2.com
   DestinationPort=1414
   ServletClient=true
   HTTP=true
   HTTPProxy=10.9.1.3
   HTTPProxyPort=80
   HTTPServer=9.100.6.7
   HTTPServerPort=8080
   ```

6. Start MQIPT1

   Open a command prompt and enter the following:

   ```
   c:
   cd \mqipt\bin
   mqipt ..
   ```

   The following message indicates successful completion:

   ```
   5639-L92 (C) Copyright IBM Corp. 2000, 2003 All Rights Reserved
   MQCPI001 Websphere MQ internet pass-thru Version 1.3.0 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 HTTP
   MQCPI024 ....and HTTP proxy at 10.9.1.3(80)
   MQCPI066 ....and HTTP server at 9.100.6.7(8080)
   MQCPI059 ....servlet client enabled
   MQCPI078 Route 1415 ready for connection requests
   ```

7. At a command prompt on the WebSphere MQ client machine, enter the following:

   ```
   SET MQSERVER=MQIPT.CONN.CHANNEL/tcp/10.9.1.2(1415)
   ```

8. Put a message using:

   ```
   amqsputc MQIPT.LOCAL.QUEUE MQIPT1.QM1
   Hello world <enter>
    <enter>
   ```

9. Get the message using:

```
amqsgetc MQIPT.LOCAL.QUEUE MQIPT1.QM1
```

You will see ″Hello world″.

HTTPS configuration

In addition to "Assumptions" on page 93, this example also makes the following assumptions:

- The Tomcat Application Server has been installed in the following directory:

  ```
  c:\jakarta-tomcat-4.0.1
  ```

  You can download Tomcat from:

  ```
  http://jakarta.apache.org/builds/jakarta-tomcat-4.0/release/v4.0.3/
  ```

- IBM Web Traffic Express has been installed in:

  ```
  c:\wte
  ```

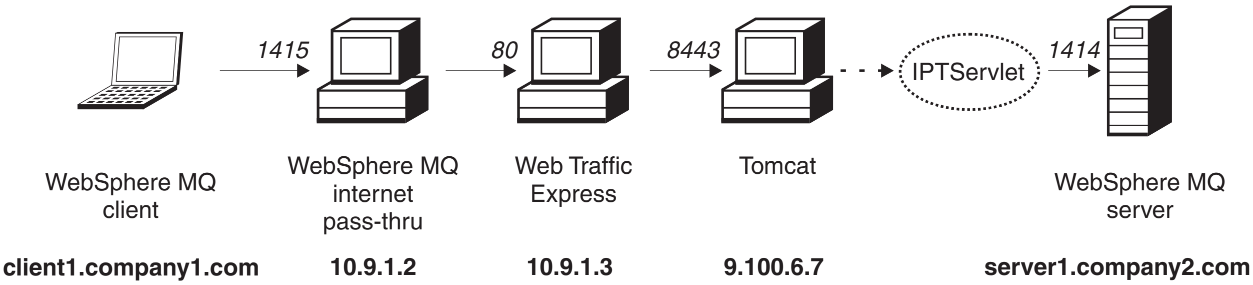
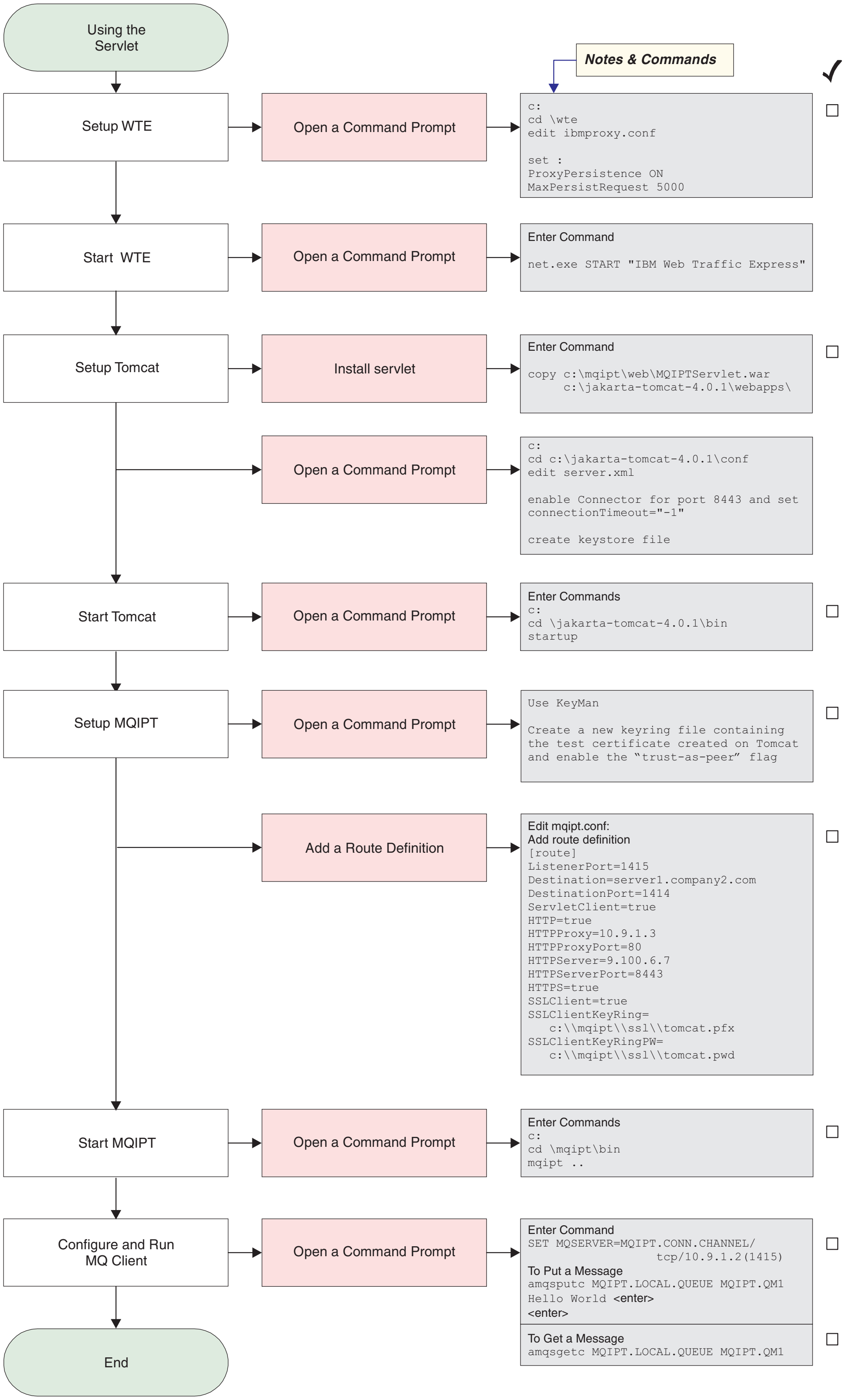


Figure 28. HTTPS network diagram



□

```
c:
cd \wte
edit ibmproxy.conf

set :
ProxyPersistence ON
MaxPersistRequest 5000
```

Enter Command

```
net.exe START "IBM Web Traffic Express"
```

□ Enter Command

```
copy c:\mqipt\web\MQIPTServlet.war
     c:\jakarta-tomcat-4.0.1\webapps\
```

```
c:
cd c:\jakarta-tomcat-4.0.1\conf
edit server.xml

enable Connector for port 8443 and set
connectionTimeout="-1"

create keystore file
```

□ Enter Commands

```
c:
cd \jakarta-tomcat-4.0.1\bin
startup
```

□

```
Use KeyMan

Create a new keyring file containing
the test certificate created on Tomcat
and enable the "trust-as-peer" flag
```

□ Edit mqipt.conf:
Add route definition

```
[route]
ListenerPort=1415
Destination=server1.company2.com
DestinationPort=1414
ServletClient=true
HTTP=true
HTTPProxy=10.9.1.3
HTTPProxyPort=80
HTTPServer=9.100.6.7
HTTPServerPort=8443
HTTPS=true
SSLClient=true
SSLClientKeyRing=
   c:\\mqipt\\ssl\\tomcat.pfx
SSLClientKeyRingPW=
   c:\\mqipt\\ssl\\tomcat.pwd
```

□ Enter Commands

```
c:
cd \mqipt\bin
mqipt ..
```

□ Enter Command

```
SET MQSERVER=MQIPT.CONN.CHANNEL/
                   tcp/10.9.1.2(1415)
```

To Put a Message

```
amqsputc MQIPT.LOCAL.QUEUE MQIPT.QM1
Hello World <enter>
<enter>
```

□ To Get a Message

```
amqsgetc MQIPT.LOCAL.QUEUE MQIPT.QM1
```

Figure 29. HTTPS configuration

1. Setup Web Traffic Express

   Edit `c:\wte\ibmroxy.conf` and set the following properties:

```
ProxyPersistence ON
MaxPersistRequest  5000
```

2. Start Web Traffic Express

   Open a command prompt and enter the following:

   ```
   net.exe Start "IBM Web Traffic Express"
   ```

3. Setup Tomcat

   To install the Servlet, copy:

   ```
   c:\mqipt\web\MQIPTServlet.war
   ```

   to:

   ```
   c:\jakarta-tomcat-4.0.1\webapps
   ```

   Edit c:\jakarta-tomcat-4.0.1\conf\server.xml, enable the connector for port 8443 and set the ConnectionTimeout property to -1.

   Use the Tomcat documentation, that is available from:

   ```
   http://jakarta.apache.org/tomcat/tomcat-4.0-doc/index.html
   ```

   and follow the instructions in ″SSL Configuration HOW-TO″ to enable SSL connections on port 8443. Create a keyring file containing a test self-signed certificate this will create a file called C:\winnt\profiles\<userid>\.keystore.

4. Start Tomcat

   Open a command prompt and enter the following:

   ```
   c:
   cd \jakarta-tomcat-4.0.1\bin
   startup
   ```

5. Copy the new keystore file from the Tomcat machine to the MQIPT machine. Use KeyMan, open the new keystore file (default password is changeit) and turn on the ″trust-as-peer″ flag (see “Creating SSL test certificates” on page 112 for more information). Save this file as c:\mqipt\ssl\tomcat.pfx and create a text file called c:\mqipt\ssl\tomcat.pwd containing the password changeit.

6. Setup MQIPT1

   Edit mqipt.conf and add a route definition:

   ```
   [route]
   ListenerPort=1415
   Destination=server1.company2.com
   DestinationPort=1414
   ServletClient=true
   HTTP=true
   HTTPProxy=10.9.1.3
   HTTPProxyPort=80
   HTTPServer=9.100.6.7
   HTTPServerPort=8443
   HTTPS=true
   SSLClient=true
   SSLClientKeyRing=c:\\mqipt\\ssl\\tomcat.pfx
   SSLClientKeyRingPW=c:\\mqipt\\ssl\\tomcat.pwd
   ```

7. Start MQIPT1

   Open a command prompt and enter the following:

   ```
   c:
   cd \mqipt\bin
   mqipt ..
   ```

   The following message indicates successful completion:

```
5639-L92 (C) Copyright IBM Corp. 2000, 2003 All Rights Reserved
MQCPI001 Websphere MQ internet pass-thru Version 1.3.0 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 HTTP
MQCPI024 ....and HTTP proxy at 10.9.1.3(80)
MQCPI066 ....and HTTP server at 9.100.6.7(8080)
MQCPI059 ....servlet client enabled
MQCPI036 ....SSL Client side enabled with properties :
MQCPI031 ......cipher suites <null>
MQCPI032 ......keyring file c:\mqipt\ssl\tomcat.pfx
MQCPI047 ......CA keyring file <null>
MQCPI038 ......distinguished name(s) CN=* O=* OU=* L=* ST=* C=*
MQCPI078 Route 1415 ready for connection requests
```

8. At a command prompt on the WebSphere MQ client machine, enter the following:

   ```
   SET MQSERVER=MQIPT.CONN.CHANNEL/tcp/10.9.1.2(1415)
   ```

9. Put a message using:

   ```
   amqsputc MQIPT.LOCAL.QUEUE MQIPT1.QM1
   Hello world <enter>
    <enter>
   ```

10. Get the message using:

    ```
    amqsgetc MQIPT.LOCAL.QUEUE MQIPT1.QM1
    ```

You will see ″Hello world″.

Configuring MQIPT Clustering support

For this example, in addition to the “Assumptions” on page 93, you must also have done the following:

On the WebSphere 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
- Socksified the queue manager

On the WebSphere 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
- Socksified the queue manager

To socksify the queue manager, either socksify whole machine or just the WebSphere MQ server application. Configure the SOCKS client to

- Point to MQIPT as the SOCKS proxy
- Enable SOCKS V5 support
- Disable user authentication
- Only make remote connections to the MQIPT

Only one application can listen on a given port address on the same machine, if port 1414 is already in use, choose a free port address and substitute it in the examples. Once you have done this you can test the routes between the queue managers by putting a message on the local queue on LONDON and retrieving it from NEWYORK.

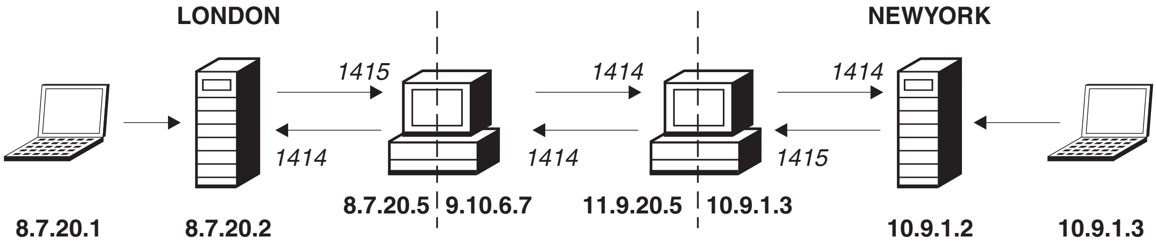
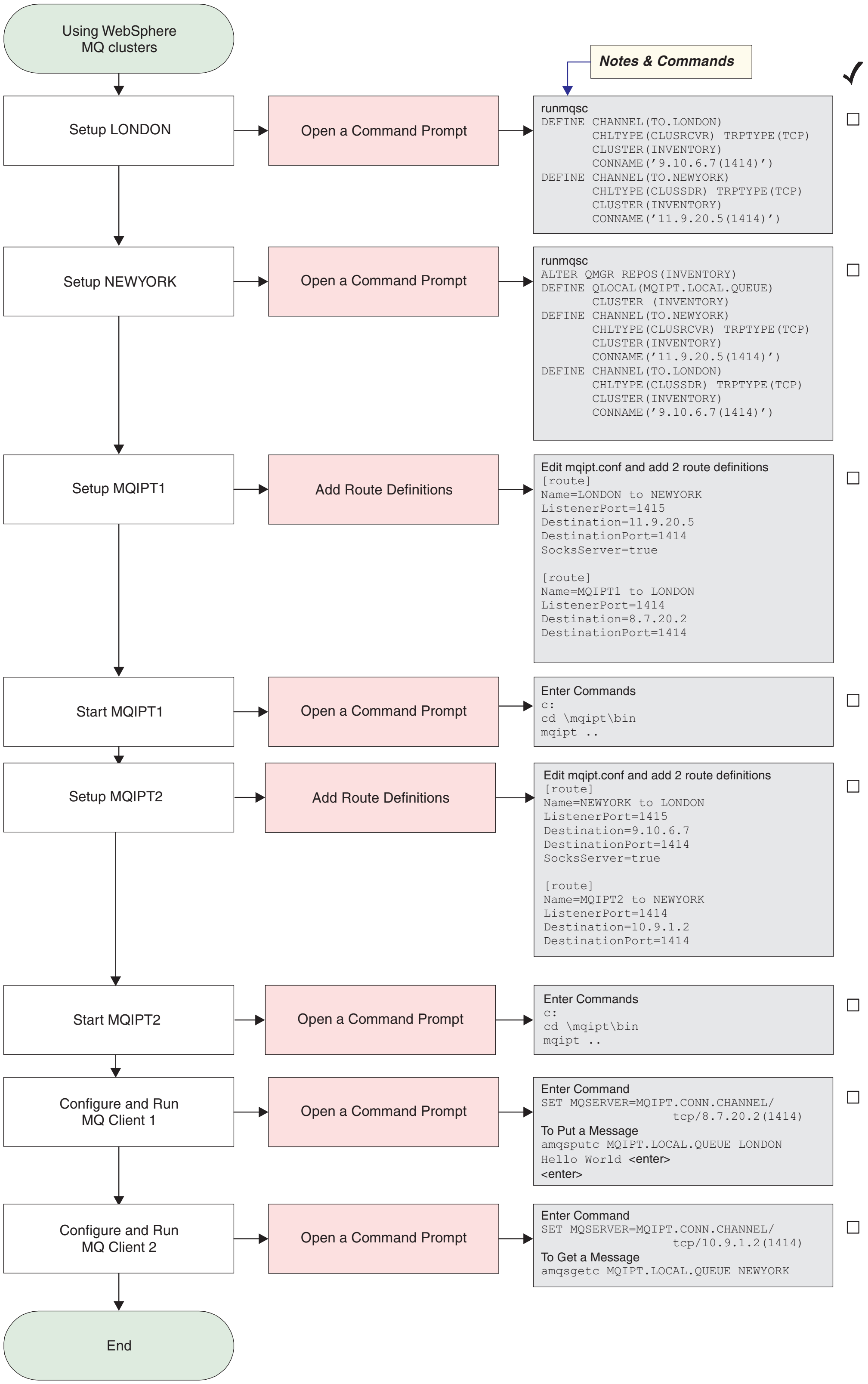


Figure 30. Clustering network diagram



□ runmqsc

```
DEFINE CHANNEL(TO.LONDON)
       CHLTYPE(CLUSRCVR) TRPTYPE(TCP)
       CLUSTER(INVENTORY)
       CONNAME('9.10.6.7(1414)')
DEFINE CHANNEL(TO.NEWYORK)
       CHLTYPE(CLUSSDR) TRPTYPE(TCP)
       CLUSTER(INVENTORY)
       CONNAME('11.9.20.5(1414)')
```

□ runmqsc

```
ALTER QMGR REPOS(INVENTORY)
DEFINE QLOCAL(MQIPT.LOCAL.QUEUE)
       CLUSTER (INVENTORY)
DEFINE CHANNEL(TO.NEWYORK)
       CHLTYPE(CLUSRCVR) TRPTYPE(TCP)
       CLUSTER(INVENTORY)
       CONNAME('11.9.20.5(1414)')
DEFINE CHANNEL(TO.LONDON)
       CHLTYPE(CLUSSDR) TRPTYPE(TCP)
       CLUSTER(INVENTORY)
       CONNAME('9.10.6.7(1414)')
```

□ Edit mqipt.conf and add 2 route definitions

```
[route]
Name=LONDON to NEWYORK
ListenerPort=1415
Destination=11.9.20.5
DestinationPort=1414
SocksServer=true

[route]
Name=MQIPT1 to LONDON
ListenerPort=1414
Destination=8.7.20.2
DestinationPort=1414
```

□ Enter Commands

```
c:
cd \mqipt\bin
mqipt ..
```

□ Edit mqipt.conf and add 2 route definitions

```
[route]
Name=NEWYORK to LONDON
ListenerPort=1415
Destination=9.10.6.7
DestinationPort=1414
SocksServer=true

[route]
Name=MQIPT2 to NEWYORK
ListenerPort=1414
Destination=10.9.1.2
DestinationPort=1414
```

□ Enter Commands

```
c:
cd \mqipt\bin
mqipt ..
```

□ Enter Command

```
SET MQSERVER=MQIPT.CONN.CHANNEL/
                  tcp/8.7.20.2(1414)
```

To Put a Message

```
amqsputc MQIPT.LOCAL.QUEUE LONDON
Hello World <enter>
<enter>
```

□ Enter Command

```
SET MQSERVER=MQIPT.CONN.CHANNEL/
                  tcp/10.9.1.2(1414)
```

To Get a Message

```
amqsgetc MQIPT.LOCAL.QUEUE NEWYORK
```

Figure 31. Clustering configuration

1. Setup LONDON

   Open a command prompt and enter the following:

```
runmqsc
DEFINE CHANNEL(TO.LONDON) +
       CHLTYPE(CLUSRCVR) TRPTYPE(TCP) +
       CLUSTER(INVENTORY) +
       CONNAME('9.10.6.7(1414)')
DEFINE CHANNEL(TO.NEWYORK) +
       CHLTYPE(CLUSSDR) TRPTYPE(TCP) +
       CLUSTER(INVENTORY) +
       CONNAME('11.9.20.5(1414)')
```

2. Setup NEWYORK

   Open a command prompt and enter the following:

   ```
   runmqsc
   ALTER QMGR REPOS(INVENTORY)
   DEFINE QLOCAL(MQIPT.LOCAL.QUEUE) +
          CLUSTER(INVENTORY)
   DEFINE CHANNEL(TO.NEWYORK) +
          CHLTYPE(CLUSRCVR) TRPTYPE(TCP) +
          CLUSTER(INVENTORY) +
          CONNAME('11.9.20.5(1414)')
   DEFINE CHANNEL(TO.LONDON) +
          CHLTYPE(CLUSSDR) TRPTYPE(TCP) +
          CLUSTER(INVENTORY) +
          CONNAME('9.10.6.7(1414)')
   ```

3. Setup MQIPT1

   Edit mqipt.conf and add two route definitions:

   ```
   [route]
   Name=LONDON to NEWYORK
   ListenerPort=1415
   Destination=11.9.20.5
   DestinationPort=1414
   SocksServer=true
   ```

   ```
   [route]
   Name=MQIPT1 to LONDON
   ListenerPort=1414
   Destination=8.7.20.2
   DestinationPort=1414
   ```

4. Start MQIPT1

   Open a command prompt and enter the following:

   ```
   c:
   cd \mqipt\bin
   mqipt ..
   ```

   The following message indicates successful completion:

   ```
   5639-L92 (C) Copyright IBM Corp. 2000, 2003 All Rights Reserved
   MQCPI001 WebSphere MQ internet pass-thru Version 1.3.0 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 ....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. Setup MQIPT2

   Edit mqipt.conf and add two route definitions:

```
[route]
Name=NEWYORK to LONDON
ListenerPort=1415
Destination=9.10.6.7
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:

   ```
   c:
   cd \mqipt\bin
   mqipt ..
   ```

   The following message indicates successful completion:

   ```
   5639-L92 (C) Copyright IBM Corp. 2000, 2003 All Rights Reserved
   MQCPI001 WebSphere MQ internet pass-thru Version 1.3.0 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 ....9.10.6.7(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 first WebSphere MQ client machine (8.7.20.1), enter the following:

   ```
   SET MQSERVER=MQIPT.CONN.CHANNEL/TCP/8.7.20.2(1414)
   ```

8. Put a message using:

   ```
   amqsputc MQIPT.LOCAL.QUEUE LONDON
   Hello world <enter>
    <enter>
   ```

9. At a command prompt on the second WebSphere MQ client machine (10.9.1.3), enter the following:

   ```
   SET MQSERVER=MQIPT.CONN.CHANNEL/TCP/10.9.1.2(1414)
   ```

10. On the second WebSphere MQ client machine, get the message using:

    ```
    amqsgetc MQIPT.LOCAL.QUEUE NEWYORK
    ```

You will see ″Hello world″.

Creating a key ring file

This sample assumes you have requested a new certificate from a trusted CA using Keyman and your personal certificate has been returned to you in a file (for example, `server.cer`). This will be sufficient to perform server authentication. If you require client authentication you will need to request a second certificate (for example, `client.cer`) and perform the following steps twice, to create two key ring files.

1. Start KeyMan
2. Select ″Create new...″
3. Select ″PKCS#12 Token″

4. Select "Action -> Generate Key"

   A new key pair will appear in the list "RSA / 1024-bit"
5. Select the new key pair
6. Select "Action -> Request Certificate"

   Following online instructions
7. Select "File -> Save"
8. Enter password
9. Enter file name of new key ring file

   For example, `c:\mqipt\ssl\myServer.pfx`
10. Keep "File format as PKCS#12 / PFX" - **do not check** "`Wrap key ring into a Java class`"
11. Select "File -> Exit"
12. Create a text file containing the passphrase (myPassWord) you used above.

    For example, `c:\mqipt\ssl\myServer.pwd`

When you get your certificate back, open the original key ring file (`myServer.pfx`). Then:

1. Start KeyMan
2. Select "Open existing...".
3. Select "Local resource"
4. Select "Open a file..."
5. Enter file name of personal cert file

   For example, `c:\mqipt\ssl\myServer.pfx`
6. Enter passphrase
7. Select "File -> Import"
8. Select "Local resource"
9. Select "Open a file..."
10. Enter server.cer

    You will see a dialog explaining the private certificate will be joined with the private key
11. Select "File -> Save"
12. Select "File -> Exit"

Repeat these steps to create a `myClient.pfx` from the `client.cer` file. Check the contents of the sample CA key ring file, `sslCAdefault.pfx`, using KeyMan, to see if your personal certificates were signed by one of the listed CA's. If this is true, then you can use the sample CA key ring file. If not, you will need to create a key ring file containing the public CA certificate that signed your personal certificates. This may have been returned with your personal certificate. If not, then you will need to request the CA certificate from the same CA that supplied your personal certificates and import it into `sslCAdefault.pfx`. The CA key ring file can be used on both client and server side. To use these new key ring files for server authentication, see the example "SSL server authentication" on page 96, 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 example "SSL client authentication" on page 98, 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
```

Allocating port addresses

This example shows how to control the local port addresses used when making outgoing connections. For this example we assume that you have installed MQIPT on a multihomed machine.

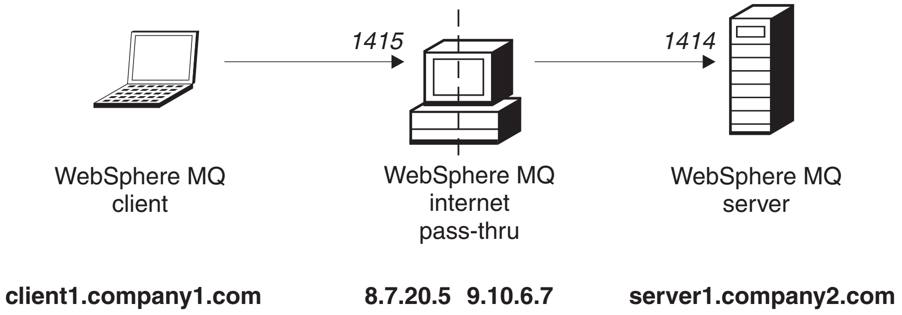
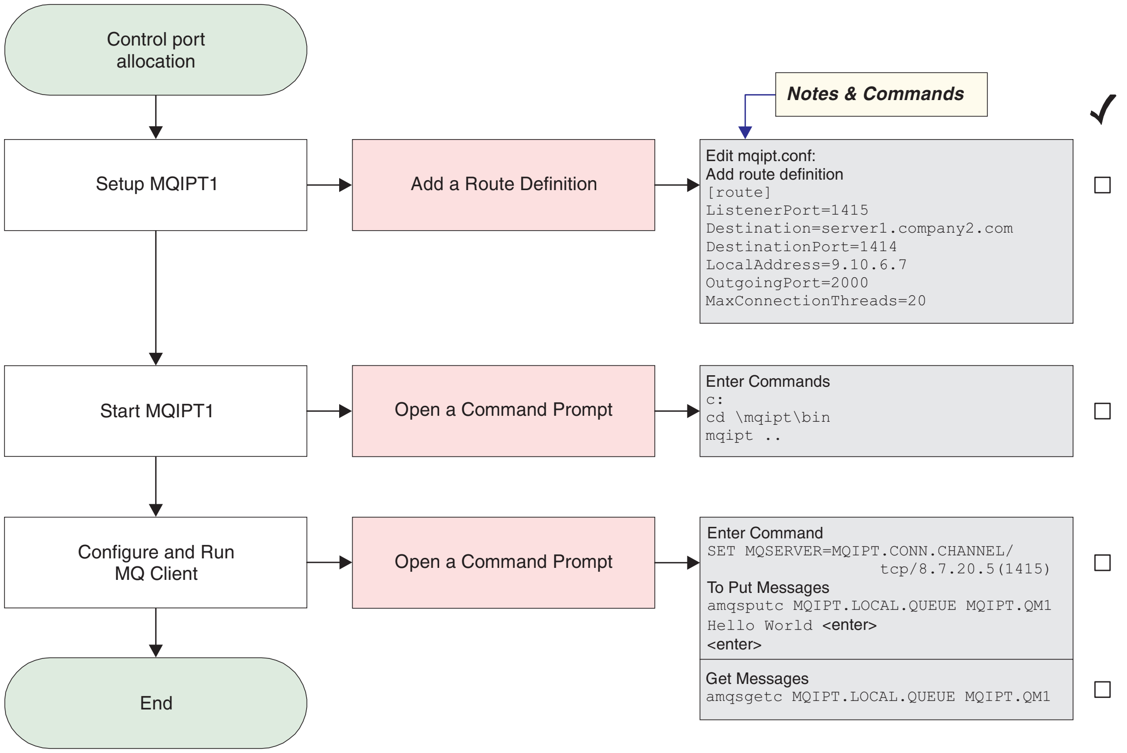


Figure 32. Port allocation network diagram



Edit mqipt.conf:
Add route definition

```
[route]
ListenerPort=1415
Destination=server1.company2.com
DestinationPort=1414
LocalAddress=9.10.6.7
OutgoingPort=2000
MaxConnectionThreads=20
```

Enter Commands

```
c:
cd \mqipt\bin
mqipt ..
```

Enter Command

```
SET MQSERVER=MQIPT.CONN.CHANNEL/
                        tcp/8.7.20.5(1415)
```

To Put Messages

```
amqsputc MQIPT.LOCAL.QUEUE MQIPT.QM1
Hello World <enter>
<enter>
```

Get Messages

```
amqsgetc MQIPT.LOCAL.QUEUE MQIPT.QM1
```

Figure 33. Port allocation configuration

1. Setup MQIPT1

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

   ```
   [route]
   ListenerPort=1415
   Destination=server1.company2.com
   DestinationPort=1414
   LocalAddress=9.10.6.7
   OutgoingPort=2000
   MaxConnectionThreads=20
   ```

2. Start MQIPT1

   Open a command prompt and enter the following:

   ```
   c:
   cd \mqipt\bin
   mqipt ..
   ```

   The following message indicates successful completion:

   ```
   5639-L92 (C) Copyright IBM Corp. 2000, 2003 All Rights Reserved
   MQCPI001 WebSphere MQ internet pass-thru Version 1.3.0 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
   MQCPI069 ....binding to local address 9.10.6.7
   MQCPI070 ....using local port address range 2000-2019
   MQCPI078 Route 1415 ready for connection requests
   ```

3. At a command prompt on the WebSphere MQ client machine, enter the following:

   ```
   SET MQSERVER=MQIPT.CONN.CHANNEL/tcp/8.7.20.5(1415)
   ```

4. Put a message using:

   ```
   amqsputc MQIPT.LOCAL.QUEUE MQIPT1.QM1
   Hello world <enter>
    <enter>
   ```

5. Get the message using:

   ```
   amqsgetc MQIPT.LOCAL.QUEUE MQIPT1.QM1
   ```

You will see ″Hello world″.

Using an LDAP server

This sample shows how to configure MQIPT to use an LDAP server to retrieve CRLs. It is not the intention of this sample to explain how to install and set up an LDAP server or 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 Certificate Authority (CA). A backup LDAP server is not being used, but could easily be implemented by adding the appropriate Route properties.

For this example, we make the following assumptions:

- IPT2 has a personal certificate, issued by the trusted 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.
- IPT1 has a copy of the trusted CA certificate which 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.

Running this sample will allow the WMQ client to connect to the Queue Manager (QM) and place a WMQ message on the target queue. Running an MQIPT trace on IPT1 will show the LDAP server being used but, to demonstrate how CRLs work, the personal certificate used by IPT2 needs to be revoked by the trusted CA. Then, in this case, the WMQ client will not be allowed to connect to the QM, as the connection from IPT1 to IPT2 will be rejected.

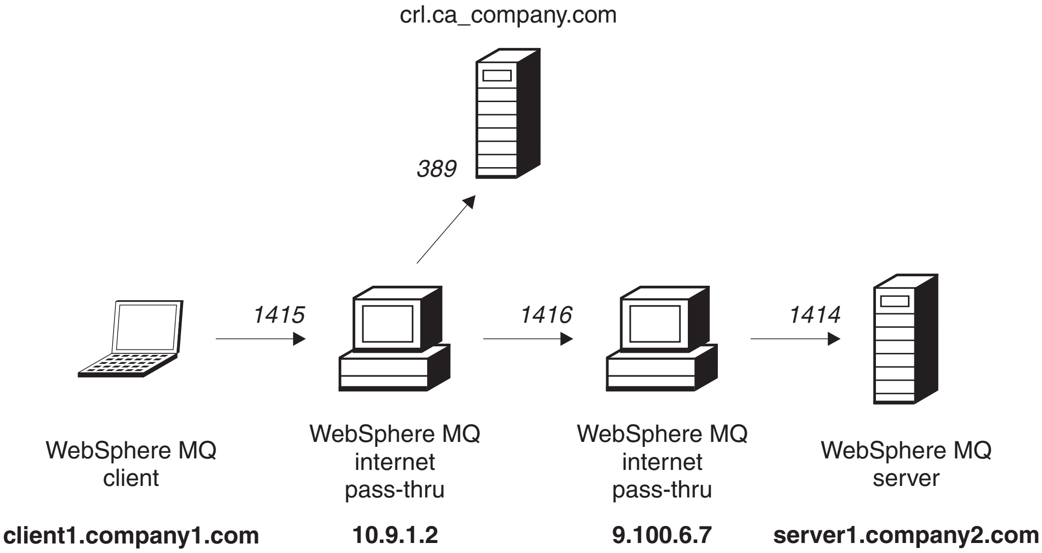
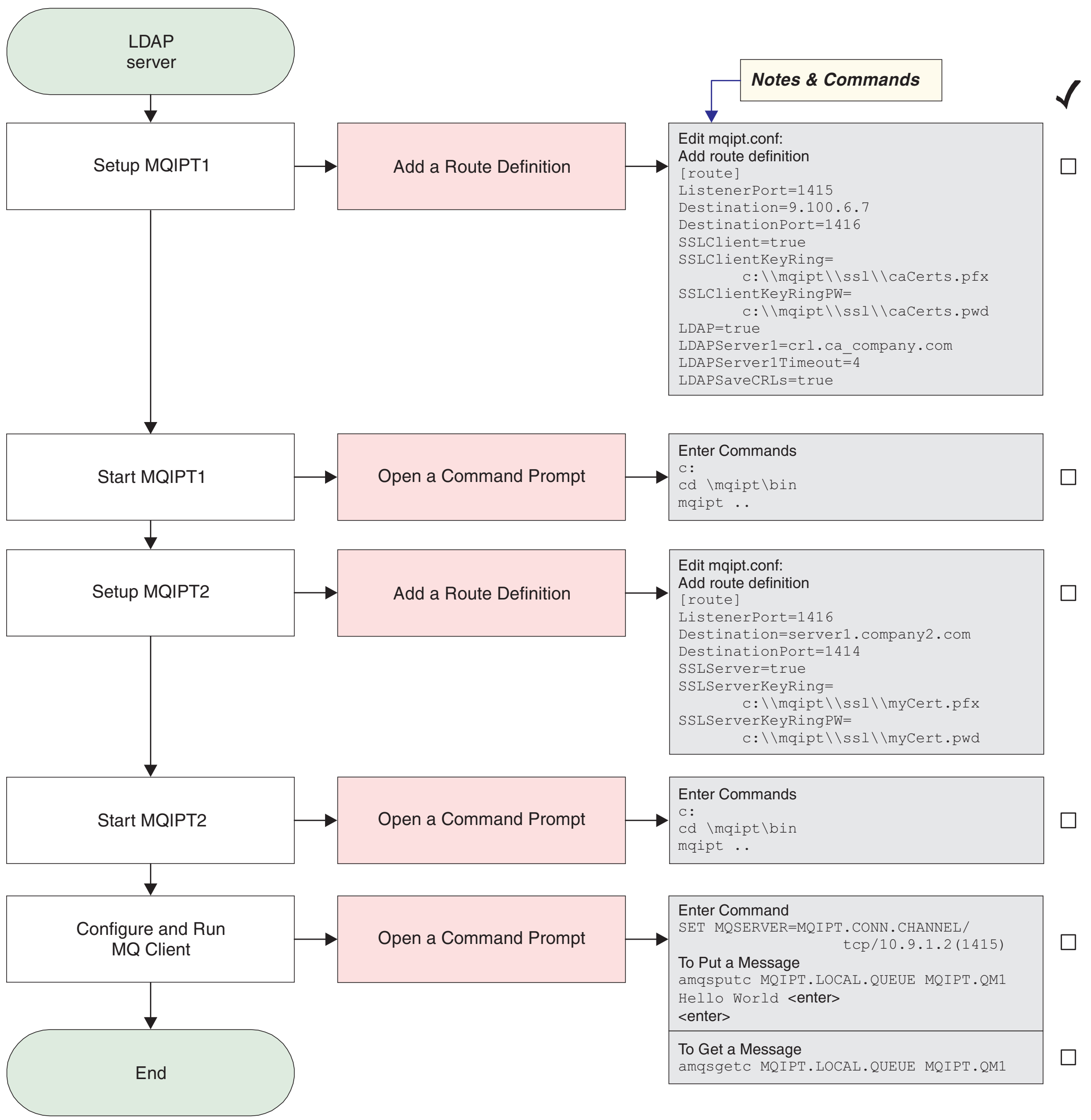


Figure 34. LDAP server network diagram



Edit mqipt.conf:
Add route definition

```
[route]
ListenerPort=1415
Destination=9.100.6.7
DestinationPort=1416
SSLClient=true
SSLClientKeyRing=
       c:\\mqipt\\ssl\\caCerts.pfx
SSLClientKeyRingPW=
       c:\\mqipt\\ssl\\caCerts.pwd
LDAP=true
LDAPServer1=crl.ca_company.com
LDAPServer1Timeout=4
LDAPSaveCRLs=true
```

Enter Commands

```
c:
cd \mqipt\bin
mqipt ..
```

Edit mqipt.conf:
Add 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
```

Enter Commands

```
c:
cd \mqipt\bin
mqipt ..
```

Enter Command

```
SET MQSERVER=MQIPT.CONN.CHANNEL/
                tcp/10.9.1.2(1415)
```

To Put a Message

```
amqsputc MQIPT.LOCAL.QUEUE MQIPT.QM1
Hello World <enter>
<enter>
```

To Get a Message

```
amqsgetc MQIPT.LOCAL.QUEUE MQIPT.QM1
```

Figure 35. LDAP server configuration

1. On IPT1

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

   ```
   [route]
   ListenerPort=1415
   Destination=9.100.6.7
   DestinationPort=1416
   SSLClient=true
   SSLClientKeyRing=c:\\mqipt\\ssl\\caCerts.pfx
   SSLClientKeyRingPW=c:\\mqipt\\ssl\\caCerts.pwd
   LDAP=true
   LDAPServer1=crl.ca_company.com
   LDAPServer1Timeout=4
   LDAPSaveCRLs=true
   ```

   Open a command prompt:

   ```
   c:
   cd \mqipt\bin
   mqipt ..
   ```

The following message indicates successful completion:

```
5639-L92 (C) Copyright IBM Corp. 2000, 2003 All Rights Reserved
MQCPI001 Websphere MQ internet pass-thru Version 1.3.0 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 ....9.100.6.7(1416)
MQCPI035 ....using MQ protocols
MQCPI036 ....SSL Client side enabled with properties :
MQCPI031 ......cipher suites <NULL>
MQCPI032 ......keyring file <NULL>
MQCPI047 ......CA keyring file c:\mqipt\ssl\caCerts.pfx
MQCPI071 ......site certificate uses CN=* O=* OU=* L=* ST=* C=*
MQCPI038 ......peer certificate uses CN=* O=* OU=* L=* ST=* C=*
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 IPT2

   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
   ```

   Open a command prompt:

   ```
   c:
   cd \mqipt\bin
   mqipt ..
   ```

   The following message indicates successful completion:

   ```
   5639-L92 (C) Copyright IBM Corp. 2000, 2003 All Rights Reserved
   MQCPI001 IBM WebSphere MQ internet pass-thru Version 1.3.0 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 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 ......cipher suites <NULL>
   MQCPI032 ......keyring file c:\mqipt\ssl\myCert.pfx
   MQCPI047 ......CA keyring file <NULL>
   MQCPI071 ......site certificate uses CN=* O=* OU=* L=* ST=* C=*
   MQCPI038 ......peer certificate uses CN=* O=* OU=* L=* ST=* C=*
   MQCPI033 ......client authentication set to false
   MQCPI078 Route 1416 ready for connection requests
   ```

3. At a command prompt on the WebSphere MQ client machine, enter the following:

   ```
   SET MQSERVER=MQIPT.CONN.CHANNEL/TCP/10.9.1.2(1415)
   ```

4. Put a message using:

   ```
   amqsputc MQIPT.LOCAL.QUEUE MQIPT.QM1
   Hello world <enter>
   <enter>
   ```

5. Get the message using:

   ```
   amqsgetc MQIPT.LOCAL.QUEUE MQIPT.QM1
   ```

You will see ″Hello world″.

SSL proxy mode

This sample shows how to run MQIPT in SSL proxy mode, so it will accept an SSL connection request from an SSL client and tunnel it to an SSL server. It assumes that the WMQ client and server are both at V5.3 and have been configured to use an SSL connection.

For further information on setting up SSL for WMQ, refer to ″WebSphere MQ Security Version 5.3″ SC34-6079-01.

For this example, we make the following assumption:

- The MQClient and QM have been set up to use SSL channel.

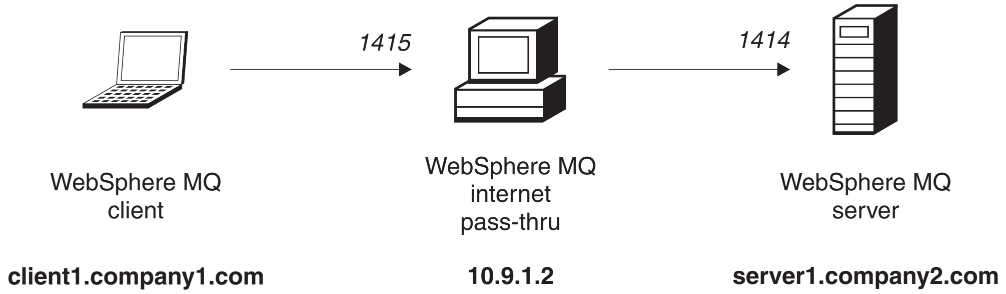
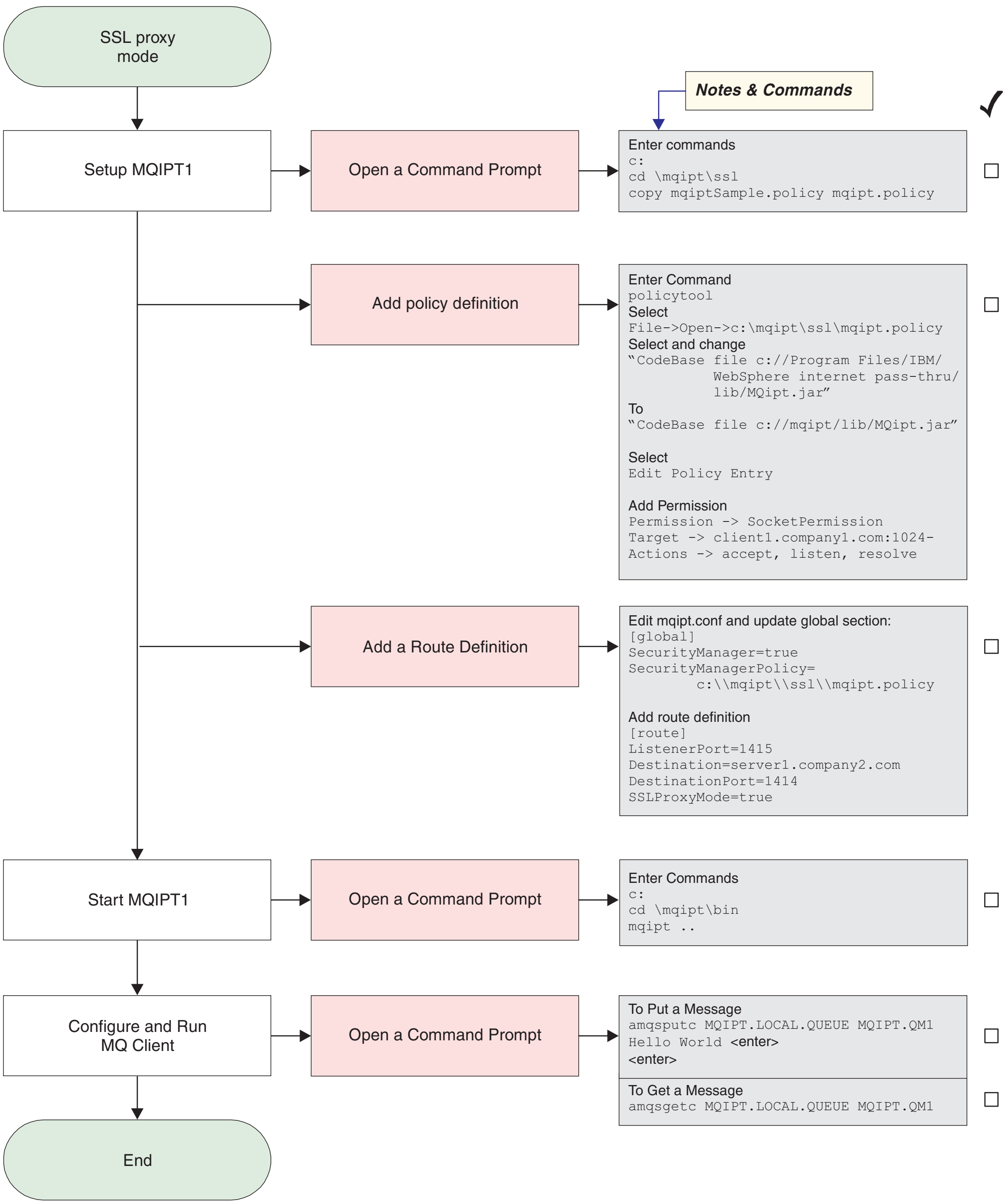


Figure 36. SSL proxy mode network diagram



Enter commands

```
c:
cd \mqipt\ssl
copy mqiptSample.policy mqipt.policy
```

Enter Command

```
policytool
```

Select

```
File->Open->c:\mqipt\ssl\mqipt.policy
```

Select and change

```
"CodeBase file c://Program Files/IBM/
          WebSphere internet pass-thru/
          lib/MQipt.jar"
```

To

```
"CodeBase file c://mqipt/lib/MQipt.jar"
```

Select

```
Edit Policy Entry
```

Add Permission

```
Permission -> SocketPermission
Target -> client1.company1.com:1024-
Actions -> accept, listen, resolve
```

Edit mqipt.conf and update global section:

```
[global]
SecurityManager=true
SecurityManagerPolicy=
        c:\\mqipt\\ssl\\mqipt.policy
```

Add route definition

```
[route]
ListenerPort=1415
Destination=server1.company2.com
DestinationPort=1414
SSLProxyMode=true
```

Enter Commands

```
c:
cd \mqipt\bin
mqipt ..
```

To Put a Message

```
amqsputc MQIPT.LOCAL.QUEUE MQIPT.QM1
Hello World <enter>
<enter>
```

To Get a Message

```
amqsgetc MQIPT.LOCAL.QUEUE MQIPT.QM1
```

Figure 37. SSL proxy mode configuration

1. On IPT1
   a. Open a command prompt and enter the following:

      `copy c:\mqipt\ssl\mqiptSample.policy to mqipt.policy`

   b. Add a policy definition using the following command:

      `policytool`

      1) Select: **File —> Open —> c:\mqipt\ssl\mqipt.policy**
      2) Select:

         `"file://C:/Program Files/IBM/WebSphere MQ internet pass-thru/lib/MQipt.jar"`

      3) Change CodeBase from:

```
"file://C:/Program Files/IBM/WebSphere MQ internet pass-thru/lib/MQipt.jar"
```

to:

```
"file://C:/mqipt/lib/MQipt.jar"
```

4) Change all permissions from:

```
"C:\\Program Files\\IBM\\WebSphere MQ internet pass-thru"
```

to:

```
"C:\\mqipt"
```

5) Add SocketPermission:

```
Permission=SocketPermission
Target = "client1.company1.com:1024-"
Actions = "accept, listen, resolve"
```

2. Edit mqipt.conf and add the following two properties to the global section and a route definition:

```
[global]
SecurityManager=true
SecurityManagerPolicy=c:\\mqipt\\ssl\\mqipt.policy

[route]
ListenerPort=1415
Destination=server1.company2.com
DestinationPort=1414
SSLProxyMode=true
```

3. Open a command prompt:

```
c:
cd \mqipt\bin
mqipt ..
```

The following message indicates successful completion:

```
5639-L92 (C) Copyright IBM Corp. 2000, 2003 All Rights Reserved
MQCPI001 Websphere MQ internet pass-thru Version 1.3.0 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 SSLProxyMode
MQCPI078 Route 1415 ready for connection requests
```

4. Put a message using:

```
amqsputc MQIPT.LOCAL.QUEUE MQIPT.QM1
Hello world <enter>
<enter>
```

5. Get the message using:

```
amqsgetc MQIPT.LOCAL.QUEUE MQIPT.QM1
```

You will see ″Hello world″.

Apache rewrite

For this example, we make the following assumptions:

- Apache HTTP server has been installed in c:\apache
- IBM Web Traffic Express has been installed in c:\wte

The sample shows how to 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.

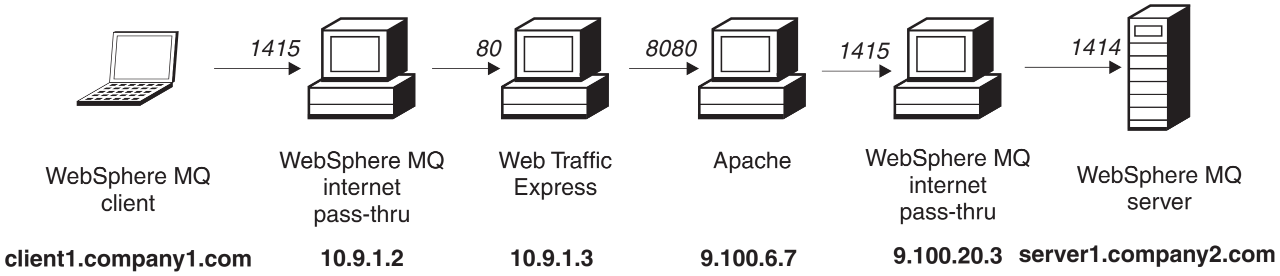
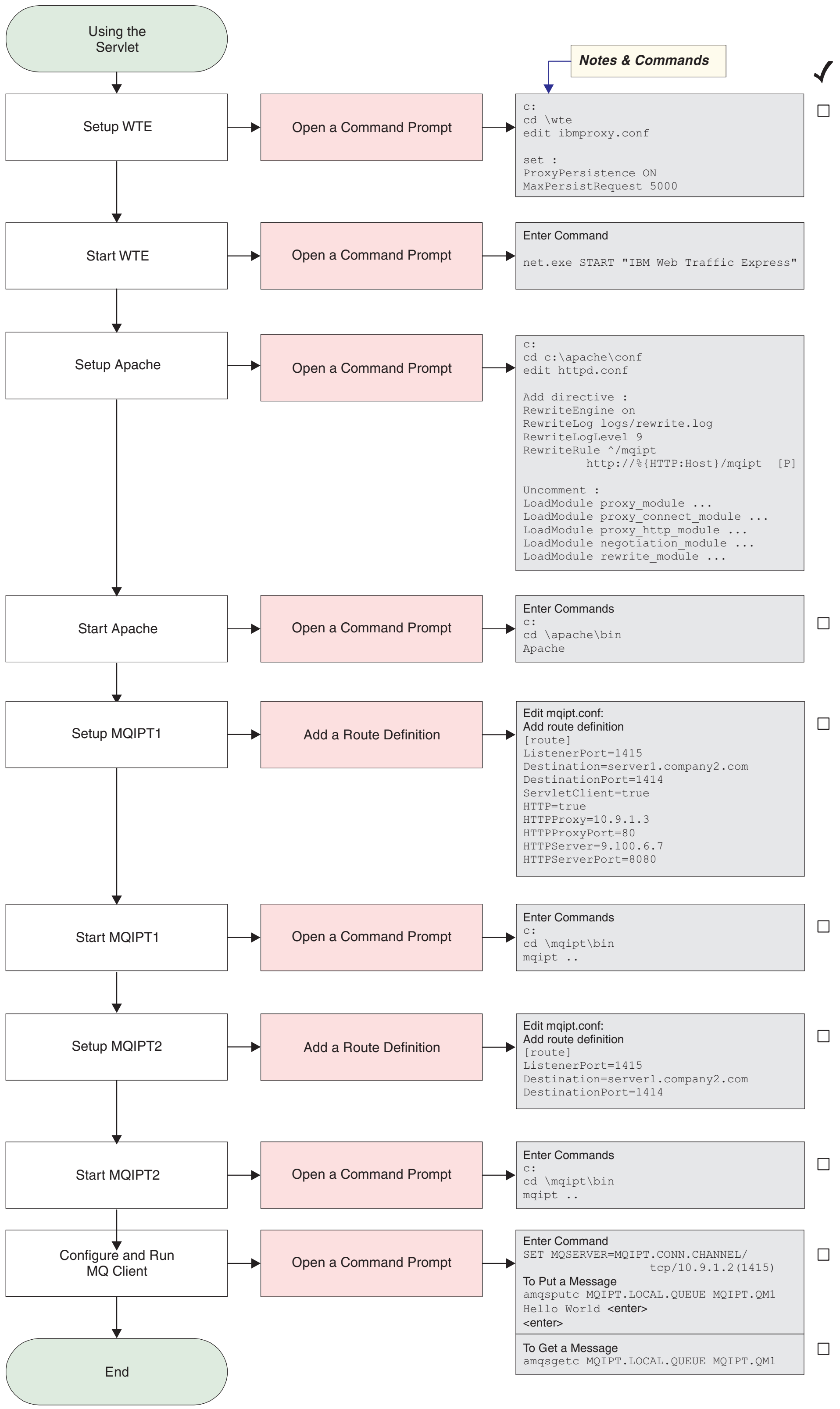


Figure 38. Apache rewrite network diagram



□

```
c:
cd \wte
edit ibmproxy.conf

set :
ProxyPersistence ON
MaxPersistRequest 5000
```

Enter Command

```
net.exe START "IBM Web Traffic Express"
```

```
c:
cd c:\apache\conf
edit httpd.conf

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

Uncomment :
LoadModule proxy_module ...
LoadModule proxy_connect_module ...
LoadModule proxy_http_module ...
LoadModule negotiation_module ...
LoadModule rewrite_module ...
```

Enter Commands □

```
c:
cd \apache\bin
Apache
```

Edit mqipt.conf:
Add route definition □

```
[route]
ListenerPort=1415
Destination=server1.company2.com
DestinationPort=1414
ServletClient=true
HTTP=true
HTTPProxy=10.9.1.3
HTTPProxyPort=80
HTTPServer=9.100.6.7
HTTPServerPort=8080
```

Enter Commands □

```
c:
cd \mqipt\bin
mqipt ..
```

Edit mqipt.conf:
Add route definition □

```
[route]
ListenerPort=1415
Destination=server1.company2.com
DestinationPort=1414
```

Enter Commands □

```
c:
cd \mqipt\bin
mqipt ..
```

Enter Command □

```
SET MQSERVER=MQIPT.CONN.CHANNEL/
                  tcp/10.9.1.2(1415)
```

To Put a Message

```
amqsputc MQIPT.LOCAL.QUEUE MQIPT.QM1
Hello World <enter>
<enter>
```

To Get a Message □

```
amqsgetc MQIPT.LOCAL.QUEUE MQIPT.QM1
```

Figure 39. Apache rewrite configuration

1. On WTE

   edit `c:\wte\ibmroxy.conf`

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 IPT1

   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=9.100.6.7
   HTTPServerPort=8080
   ```

   Open a command prompt:

   ```
   c:
   cd \mqipt\bin
   mqipt ..
   ```

   The following message indicates successful completion:

   ```
   5639-L92 (C) Copyright IBM Corp. 2000, 2003 All Rights Reserved
   MQCPI001 Websphere MQ internet pass-thru Version 1.3.0 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 HTTP
   MQCPI024 ....and HTTP proxy at 10.9.1.3(80)
   MQCPI066 ....and HTTP server at 9.100.6.7(8080)
   MQCPI078 Route 1415 ready for connection requests
   ```

4. On IPT2

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

   ```
   [route]
   ListenerPort=1415
   Destination=server1.company2.com
   DestinationPort=1414
   ```

   Open a command prompt:

   ```
   c:
   cd \mqipt\bin
   mqipt ..
   ```

   The following message indicates successful completion:

```
5639-L92 (C) Copyright IBM Corp. 2000, 2003 All Rights Reserved
MQCPI001 Websphere MQ internet pass-thru Version 1.3.0 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 WebSphere MQ client machine, enter the following:

   ```
   SET MQSERVER=MQIPT.CONN.CHANNEL/TCP/10.9.1.2(1415)
   ```

6. Put a message using:

   ```
   amqsputc MQIPT.LOCAL.QUEUE MQIPT.QM1
   Hello world <enter>
   <enter>
   ```

7. Get the message using:

   ```
   amqsgetc MQIPT.LOCAL.QUEUE MQIPT.QM1
   ```

You will see ″Hello world″.

Security exit

For this example, we make the following assumptions:

- Java 1.4 SDK installed
- The Java bin subdirectory has been added to PATH environment variable

This is a simple test to show how to use the supplied sample security exit called SampleSecurityExit. This security exit has been written to only allow client connections using a channel name starting with the characters ″MQIPT.″.

Using the suggested srvconn channel name of ″MQIPT.CONN.CHANNEL″ (as used in most of these samples), the client connection will be allowed to complete and a WMQ message can be placed on the queue.

To prove 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.

To show the ″TEST.CONN.CHANNEL″ is working without using the security exit, set the MQSERVER environment variable to point directly to the WMQ listener port (for example, 1414), so MQIPT is not being used. This time the amqsputc command will work as expected.

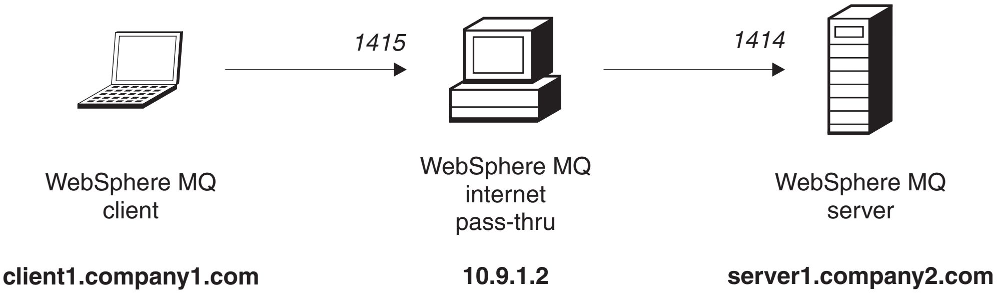
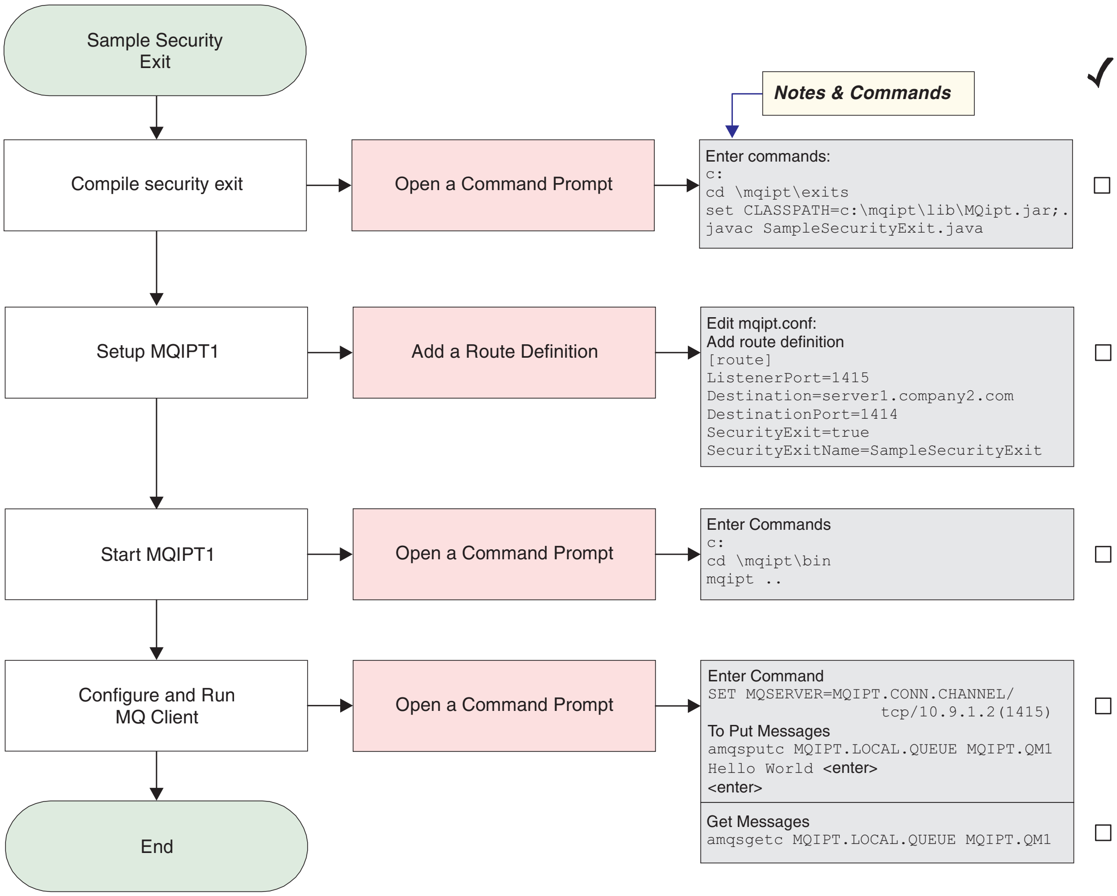


Figure 40. Security exit network diagram



```
c:
cd \mqipt\exits
set CLASSPATH=c:\mqipt\lib\MQipt.jar;.
javac SampleSecurityExit.java
```

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

```
c:
cd \mqipt\bin
mqipt ..
```

```
SET MQSERVER=MQIPT.CONN.CHANNEL/
                    tcp/10.9.1.2(1415)
```

```
amqsputc MQIPT.LOCAL.QUEUE MQIPT.QM1
Hello World <enter>
<enter>
```

```
amqsgetc MQIPT.LOCAL.QUEUE MQIPT.QM1
```

Figure 41. Security exit configuration

1. On IPT1

   Open a command prompt:

   ```
   c:
   cd \mqipt\exits
   set CLASSPATH=c:\mqipt\lib\MQipt.jar;.
   javac SampleSecurityExit.java
   ```

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

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

Open a command prompt:

```
c:
cd \mqipt\bin
mqipt ..
```

The following message indicates successful completion:

```
5639-L92 (C) Copyright IBM Corp. 2000, 2003 All Rights Reserved
MQCPI001 Websphere MQ internet pass-thru Version 1.3.0 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
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 WebSphere MQ client machine, enter the following:

   ```
   SET MQSERVER=MQIPT.CONN.CHANNEL/TCP/10.9.1.2(1415)
   ```

3. Put a message using:

   ```
   amqsputc MQIPT.LOCAL.QUEUE MQIPT.QM1
   Hello world <enter>
   <enter>
   ```

4. Get the message using:

   ```
   amqsgetc MQIPT.LOCAL.QUEUE MQIPT.QM1
   ```

You will see ″Hello world″.

Routing security exit

For this example, we make the following assumptions:

- Java 1.4 SDK installed
- The Java bin subdirectory has been added to PATH environment variable
- Three identical queue managers have been created on three separate servers

This is a working example which will dynamically route client connection requests in a round robin fashion, to a group of WMQ Queue Manager servers. The Queue Manager on each server in the group must be a mirror image of every other one.

The list of server names will be read from a configuration file. The name and location of the configuration file is defined with the SecurityExitName and SecurityExitPath properties. The sample config file, called SampleRoutingExit.conf, contains the entries :

```
server1.company.com:1414
server2.company.com:1415
server3.company.com:1416
```

You must change these server names to match your environment.

The first time the amqsputc command is issued, the WMQ message will be placed on the MQIPT.LOCAL.QUEUE on the QM on server1. The second time it is issued, the message will appear on the QM on server2, and so on. Using this setup, it is not possible for the amqsgetc command to retrieve the message just placed on the queue, as the client connection request used by the amqsgetc command will be passed to the next QM in the list. But issuing three amqsputc commands, followed by three amqsgetc commands, will ensure each message is retrieved in the same order. Of course, by using another WMQ client, connecting directly to a QM (that is, not using the MQIPT in this sample), you can selectively retrieve messages from any of the queue managers.

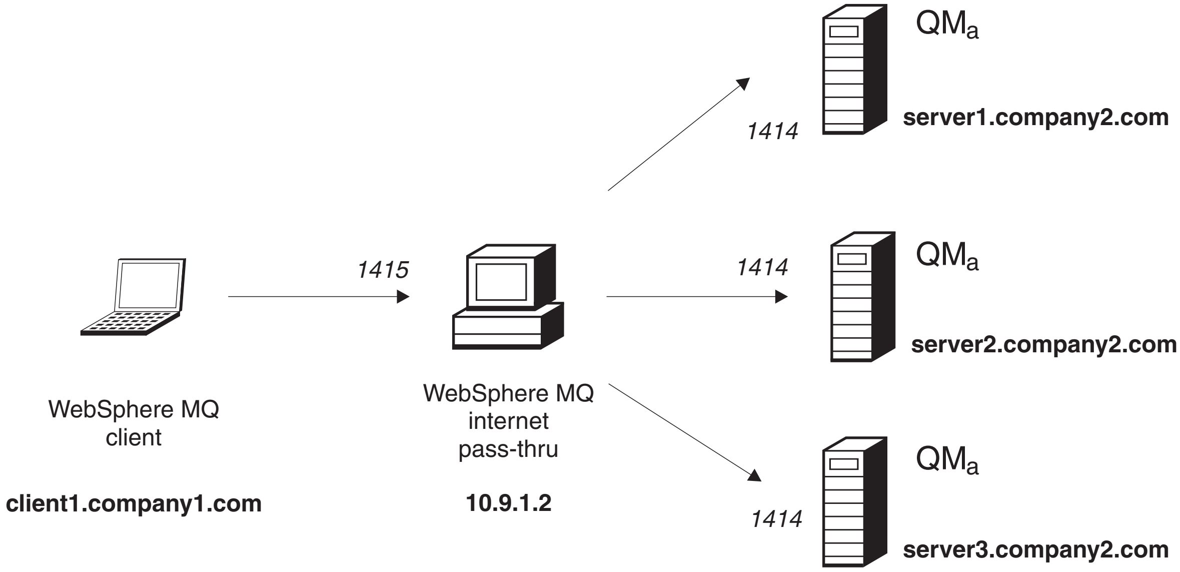
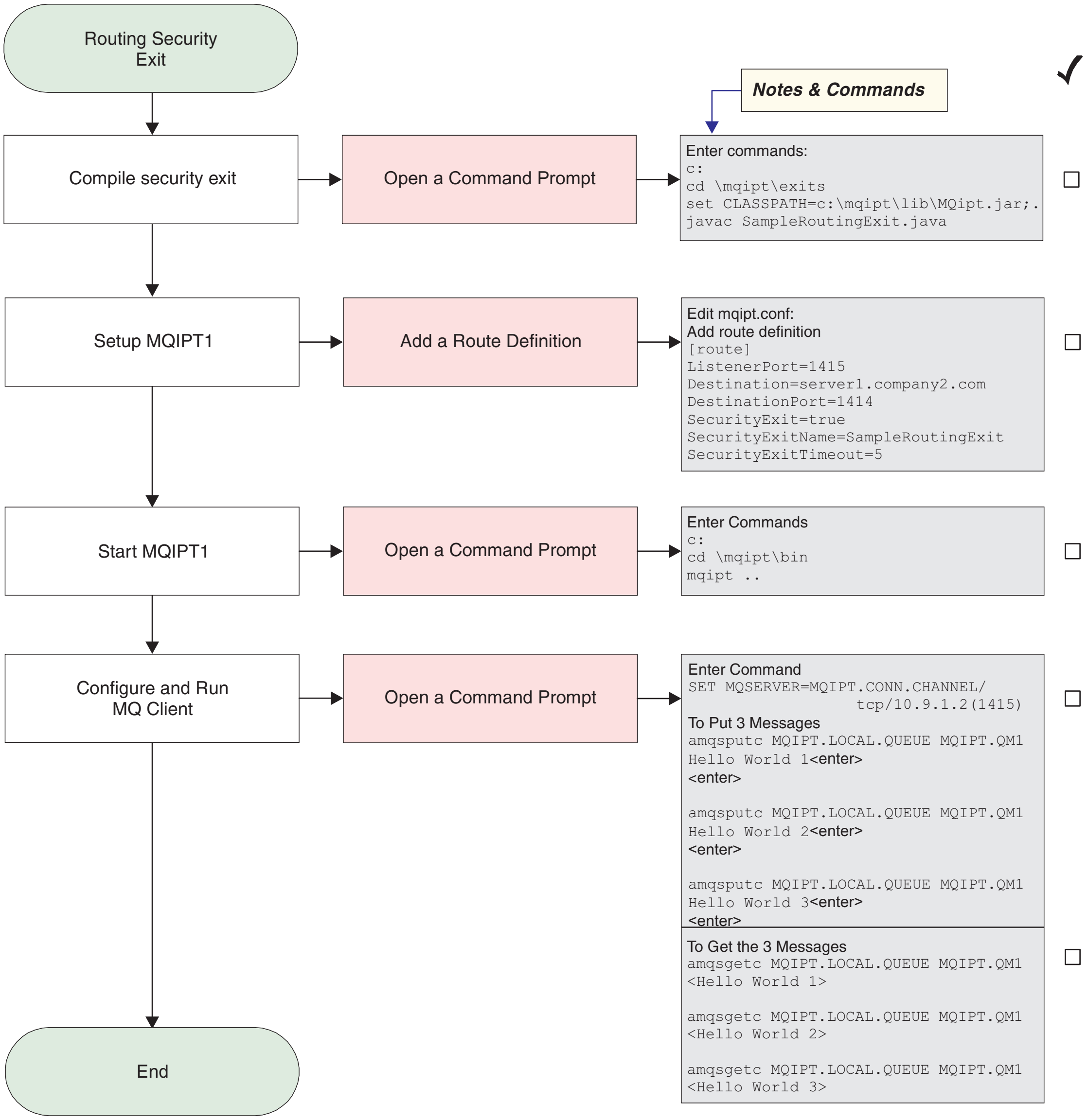


Figure 42. Routing security exit network diagram



Enter commands:

```
c:
cd \mqipt\exits
set CLASSPATH=c:\mqipt\lib\MQipt.jar;.
javac SampleRoutingExit.java
```

Edit mqipt.conf:
Add route definition

```
[route]
ListenerPort=1415
Destination=server1.company2.com
DestinationPort=1414
SecurityExit=true
SecurityExitName=SampleRoutingExit
SecurityExitTimeout=5
```

Enter Commands

```
c:
cd \mqipt\bin
mqipt ..
```

Enter Command

```
SET MQSERVER=MQIPT.CONN.CHANNEL/
                    tcp/10.9.1.2(1415)
```

To Put 3 Messages

```
amqsputc MQIPT.LOCAL.QUEUE MQIPT.QM1
Hello World 1<enter>
<enter>

amqsputc MQIPT.LOCAL.QUEUE MQIPT.QM1
Hello World 2<enter>
<enter>

amqsputc MQIPT.LOCAL.QUEUE MQIPT.QM1
Hello World 3<enter>
<enter>
```

To Get the 3 Messages

```
amqsgetc MQIPT.LOCAL.QUEUE MQIPT.QM1
<Hello World 1>

amqsgetc MQIPT.LOCAL.QUEUE MQIPT.QM1
<Hello World 2>

amqsgetc MQIPT.LOCAL.QUEUE MQIPT.QM1
<Hello World 3>
```

Figure 43. Routing security exit configuration

1. On IPT1

   Open a command prompt:

   ```
   c:
   cd \mqipt\exits
   set CLASSPATH=c:\mqipt\lib\MQipt.jar;.
   javac SampleRoutingExit.java
   ```

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

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

   Open a command prompt:

```
c:
cd \mqipt\bin
mqipt ..
```

The following message indicates successful completion:

```
5639-L92 (C) Copyright IBM Corp. 2000, 2003 All Rights Reserved
MQCPI001 Websphere MQ internet pass-thru Version 1.3.0 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
MQCPI079 ....using security exit c:\mqipt\exits\SampleRoutingExit
MQCPI080 ......and timeout of 5 seconds
MQCPI078 Route 1415 ready for connection requests
```

2. At a command prompt on the WebSphere MQ client machine, enter the following:

   ```
   SET MQSERVER=MQIPT.CONN.CHANNEL/TCP/10.9.1.2(1415)
   ```

3. Put three messages using:

   ```
   amqsputc MQIPT.LOCAL.QUEUE MQIPT.QM1
   Hello world 1 <enter>
   <enter>
   amqsputc MQIPT.LOCAL.QUEUE MQIPT.QM1
   Hello world 2 <enter>
   <enter>
   amqsputc MQIPT.LOCAL.QUEUE MQIPT.QM1
   Hello world 3 <enter>
   <enter>
   ```

4. Get the messages using:

   ```
   amqsgetc MQIPT.LOCAL.QUEUE MQIPT.QM1
   amqsgetc MQIPT.LOCAL.QUEUE MQIPT.QM1
   amqsgetc MQIPT.LOCAL.QUEUE MQIPT.QM1
   ```

You will see ″Hello world 1″, ″Hello world 2″, and ″Hello world 3″.

Dynamic one route exit

For this example, we make the following assumptions:

- Java 1.4 SDK installed
- The Java bin subdirectory has been added to PATH environment variable
- Three different queue managers have been created on three separate servers

This is a working example which shows how to dynamically route client connection requests to a target server, based on the name of the channel being used. The first part of the channel name is the name of the Queue Manager, so for example, to connect to QM1, the name of a svrconn channel would be QM1.MQIPT.CONN.CHANNEL. Using this channel naming convention, MQIPT need only use one route to service all connection requests.

The list of Queue Manager and server names will be read from a configuration file. The name and location of the configuration file is defined with the SecurityExitName and SecurityExitPath properties. The sample config file, called SampleOneRouteExit.conf, contains the entries:

```
QM1   server1.company.com:1414
QM2   server2.company.com:1415
QM3   server3.company.com:1416
```

You must change these server names to match your environment.

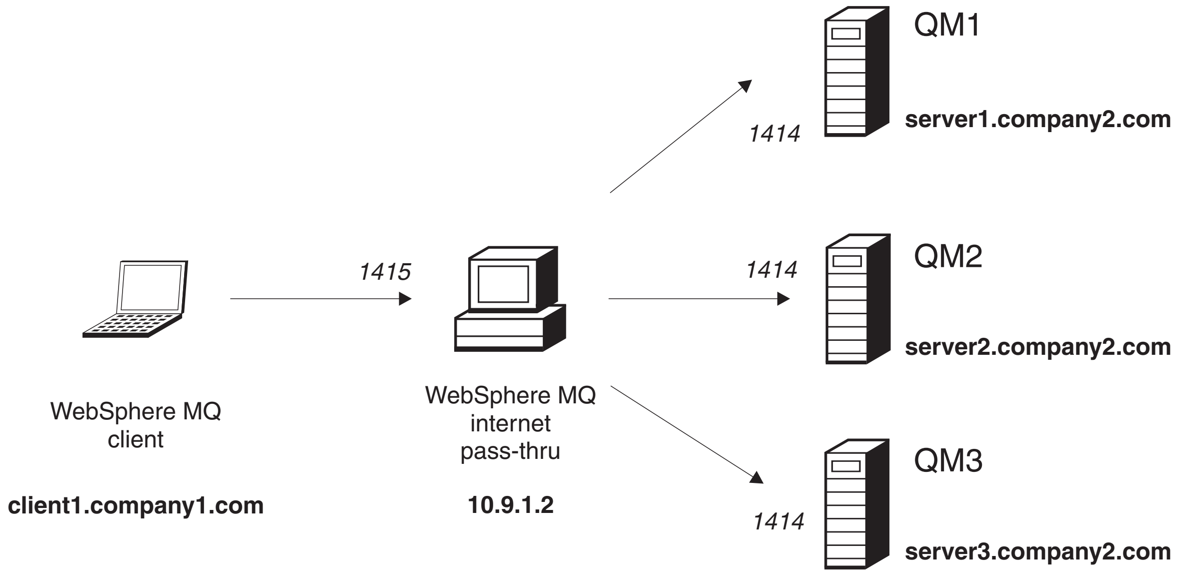
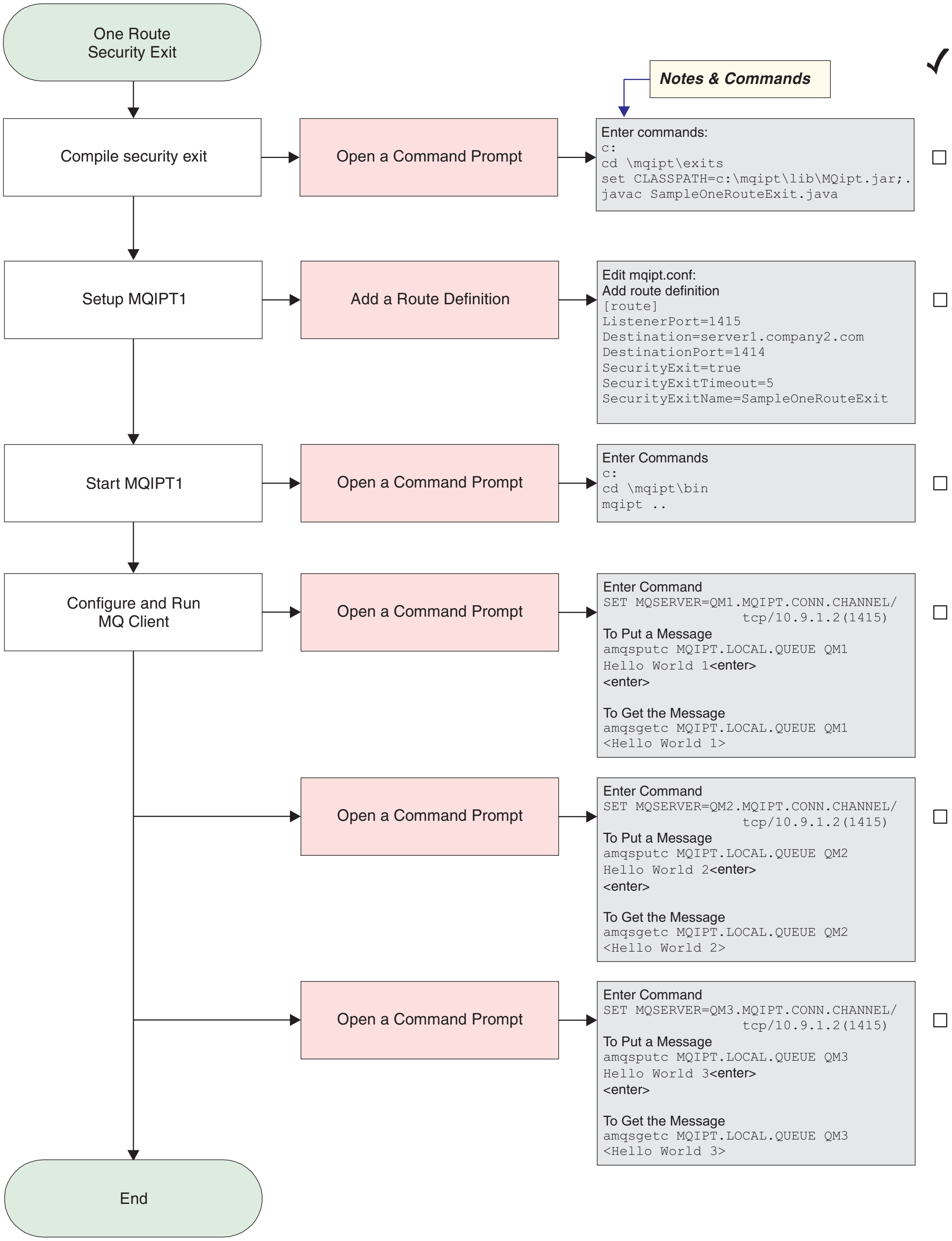


Figure 44. Dynamic one route exit network diagram



Enter commands:

```
c:
cd \mqipt\exits
set CLASSPATH=c:\mqipt\lib\MQipt.jar;.
javac SampleOneRouteExit.java
```

Edit mqipt.conf:
Add route definition

```
[route]
ListenerPort=1415
Destination=server1.company2.com
DestinationPort=1414
SecurityExit=true
SecurityExitTimeout=5
SecurityExitName=SampleOneRouteExit
```

Enter Commands

```
c:
cd \mqipt\bin
mqipt ..
```

Enter Command

```
SET MQSERVER=QM1.MQIPT.CONN.CHANNEL/
                 tcp/10.9.1.2(1415)
```

To Put a Message

```
amqsputc MQIPT.LOCAL.QUEUE QM1
Hello World 1<enter>
<enter>
```

To Get the Message

```
amqsgetc MQIPT.LOCAL.QUEUE QM1
<Hello World 1>
```

Enter Command

```
SET MQSERVER=QM2.MQIPT.CONN.CHANNEL/
                 tcp/10.9.1.2(1415)
```

To Put a Message

```
amqsputc MQIPT.LOCAL.QUEUE QM2
Hello World 2<enter>
<enter>
```

To Get the Message

```
amqsgetc MQIPT.LOCAL.QUEUE QM2
<Hello World 2>
```

Enter Command

```
SET MQSERVER=QM3.MQIPT.CONN.CHANNEL/
                 tcp/10.9.1.2(1415)
```

To Put a Message

```
amqsputc MQIPT.LOCAL.QUEUE QM3
Hello World 3<enter>
<enter>
```

To Get the Message

```
amqsgetc MQIPT.LOCAL.QUEUE QM3
<Hello World 3>
```

Figure 45. Dynamic one route exit configuration

1. On IPT1

   Open a command prompt:

   ```
   c:
   cd \mqipt\exits
   set CLASSPATH=c:\mqipt\lib\MQipt.jar;.
   javac SampleOneRouteExit.java
   ```

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

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

Open a command prompt:

```
c:
cd \mqipt\bin
mqipt ..
```

The following message indicates successful completion:

```
5639-L92 (C) Copyright IBM Corp. 2000, 2003 All Rights Reserved
MQCPI001 Websphere MQ internet pass-thru Version 1.3.0 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
MQCPI079 ....using security exit c:\mqipt\exits\SampleOneRouteExit
MQCPI080 ......and timeout of 5 seconds
MQCPI078 Route 1415 ready for connection requests
```

2. At a command prompt on the WebSphere MQ client machine, enter the following:

   ```
   SET MQSERVER=QM1.MQIPT.CONN.CHANNEL/TCP/10.9.1.2(1415)
   ```

3. Put a message using:

   ```
   amqsputc MQIPT.LOCAL.QUEUE QM1
   Hello world 1 <enter>
   <enter>
   ```

4. Get the message using:

   ```
   amqsgetc MQIPT.LOCAL.QUEUE QM1
   ```

   You will see ″Hello world 1″.

5. At a command prompt on the WebSphere MQ client machine, enter the following:

   ```
   SET MQSERVER=QM2.MQIPT.CONN.CHANNEL/TCP/10.9.1.2(1415)
   ```

6. Put a message using:

   ```
   amqsputc MQIPT.LOCAL.QUEUE QM2
   Hello world 2 <enter>
   <enter>
   ```

7. Get the message using:

   ```
   amqsgetc MQIPT.LOCAL.QUEUE QM2
   ```

   You will see ″Hello world 2″.

8. At a command prompt on the WebSphere MQ client machine, enter the following:

   ```
   SET MQSERVER=QM3.MQIPT.CONN.CHANNEL/TCP/10.9.1.2(1415)
   ```

9. Put a message using:

   ```
   amqsputc MQIPT.LOCAL.QUEUE QM3
   Hello world 3 <enter>
   <enter>
   ```

10. Get the message using:

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

    You will see ″Hello world 3″.

Chapter 21. Looking after internet pass-thru

This chapter describes how to keep internet pass-thru running, under these headings:

- "Maintenance"
- "Problem determination"
- "Performance tuning" on page 148

Maintenance

You should back up the following files on a regular basis as part of your normal backup procedures:

- The configuration file, `mqipt.conf`
- The SSL key ring files in `mqipt.conf` as defined with the following properties:
  - `SSLClientKeyRing`
  - `SSLClientCAKeyRing`
  - `SSLServerKeyRing`
  - `SSLServerCAKeyRing`
- The SSL key ring password files in `mqipt.conf` as defined with the following properties:
  - `SSLClientKeyRingPW`
  - `SSLClientCAKeyRingPW`
  - `SSLServerKeyRingPW`
  - `SSLServerCAKeyRingPW`
- The Administration Client configuration file, `client.conf`, which contains connection information about all the MQIPTs known to the Administration Client.

Problem determination

There are some common pitfalls to check first if you encounter a problem:

- The MQIPT system has just been installed and has not been rebooted.
- HTTP has been set to `true` on a route directly connected to a queue manager.
- SSLClient has been set to `true` on a route directly connected to a queue manager.
- The CLASSPATH has not been set up correctly.
- The PATH has not been set up correctly.
- The passwords stored for the key ring files are case-sensitive.

The next step is to follow the flowchart shown in Figure 46 on page 146. The numbers refer to the following notes.

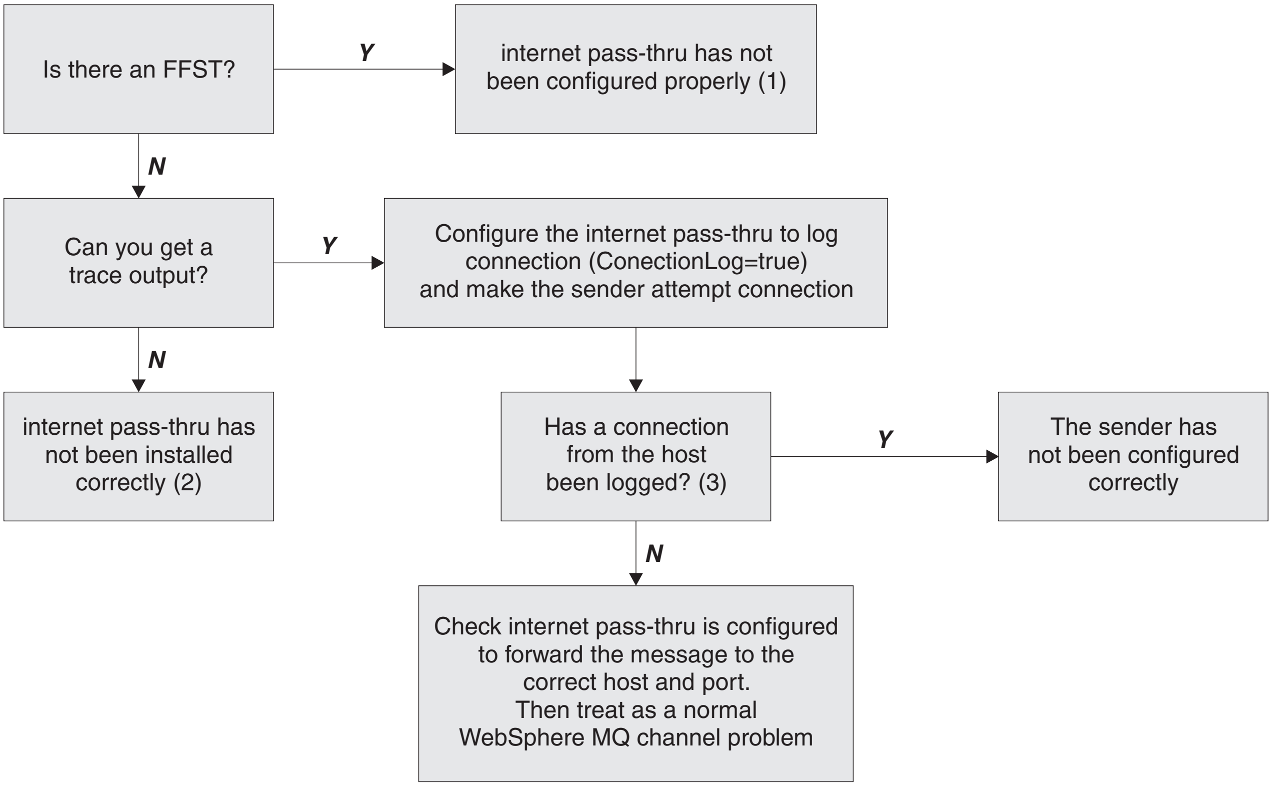

Is there an FFST?
Y
internet pass-thru has not been configured properly (1)
N
Can you get a trace output?
Y
Configure the internet pass-thru to log connection (ConectionLog=true) and make the sender attempt connection
N
internet pass-thru has not been installed correctly (2)
Has a connection from the host been logged? (3)
Y
The sender has not been configured correctly
N
Check internet pass-thru is configured to forward the message to the correct host and port. Then treat as a normal WebSphere MQ channel problem

Figure 46. Problem determination flowchart

Notes:

1. If you find any FFST reports (in the errors subdirectory), you know that MQIPT was correctly installed. 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 FFSTs and restart or refresh MQIPT.
2. If MQIPT has not been installed correctly, check that all the files have been put in the correct place and the CLASSPATH has been updated. To check this is correct, try to start MQIPT manually.
3. Manually starting MQIPT.

   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 "`..`".

   This will start MQIPT and look for the configuration in the home directory. Look for any error messages and FFSTs in the `errors` subdirectory.

   Look at the text output from MQIPT for any error messages and correct the error(s). Check for FFSTs and correct any errors. 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.

Automatically starting internet pass-thru

If you install MQIPT as a Windows NT Service, and have changed its startup to be automatic, it starts when the system is brought up. Always start MQIPT manually once before trying to install MQIPT as a Windows NT Service to confirm correct installation. See "Using a Windows service control program" on page 49 for more details.

If you receive the error message "Unable to locate DLL...", either you are using the wrong `mqiptService` program or you have not configured the system PATH environment variable correctly. PATH must contain the location of the JNI runtime libraries. This file (`jvm.dll`) can be found in the client subdirectory of the JDK.

Checking for end-to-end connectivity

If MQIPT is correctly installed, the next step is to check that the routes are set up correctly.

In the configuration file, `mqipt.conf`, set the `ConnectionLog` property to true. Start or refresh MQIPT and attempt a connection. The connect log is created in the logs directory below the home directory. If it is not created, you know that MQIPT has not been installed correctly. If no connection attempts are recorded, the sender has not been set up correctly. If attempts are recorded, check that MQIPT is forwarding the messages to the correct address.

Tracing errors

MQIPT provides a detailed execution trace facility, which is controlled by the trace attribute. Each route can be traced independently. Trace files are written to the *xxx*`\errors` directory (where *xxx* is the directory containing `mqipt.conf`. Each trace file produced has a name with the following format:

`iptroute`*nnnnn*`.trc`

where *nnnnn* is the number of the port on which the route is listening. Trace output from threads not directly associated with any particular route (for example, the thread handling command input) is written to a separate file called `iptmain.trc`.

Unexpected fatal errors are written as FFST records to an error log file, held in the *xxx*`\errors` directory (where *xxx* is the directory containing `mqipt.conf`). The FFST files have the following format:

`ipt`*xxx*`.FFST`

where *xxx* is the sequence that the FFST was generated (1 is the oldest). In a long-running system, you might reach the maximum number the system can generate. In this case, any FFSTs that are generated are written to the file `mqipt0.FFST`. If the file `mqipt0.FFST` is created, you should stop and restart MQIPT at the first opportunity and delete the old files.

Reporting problems

If you do have to report a problem to the IBM Service Center, it will help to resolve the problem more quickly if you can provide the following information:

- Provide a simple network diagram of machines being used, including IP addresses
- If there is more than one MQIPT being used, synchronize the system clock on each MQIPT machine - this will help to match trace entries in each MQIPT
- Erase old trace files

- Run the client to produce the problem - so trace files only contain one instance of the problem
- Send copy of all MQIPT .trc and .log files

Performance tuning

Here are some pointers for tuning your system.

Thread pool management

The relative performance of each route can be tuned using a combination of a thread pool and an idle timeout specification.

Connection threads

Each 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 nominated to handle the first incoming request. When this request comes in, the thread is set to work on this request immediately, and the next thread assigned as ready for the next incoming request. When all threads are assigned to work, a new thread is created, added to the working pool, and assigned for work. In this way, the pool grows until `MaxConnectionThreads` is reached. When the number of working threads is at `MaxConnectionThreads`, the next incoming request waits until a thread is released back to the working pool. This is the maximum working capacity of the route, after which no additional requests can be accepted. Threads are released back to the pool when a conversation ends, or the specified idle timeout period has elapsed.

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, an idle timeout interval may be specified, so that any thread that has been inactive for the specified period of time (in minutes) is terminated. A monitor thread keeps a regular check on thread idle times, and terminates those that have exceeded the threshold. Threads are recycled for use by placing them back into the working pool.

Chapter 22. Messages

When run from the command line, MQIPT displays a small number of information, warning, and error messages on the console.

Note that:

- MQCAxxxx messages are Administration Client messages.
- MQCPxxxx messages are MQIPT messages.
- MQCxIxxx messages are information messages.
- MQCxWxxx messages are warning messages.
- MQCxExxx messages are error messages.

MQCAE001 Unknown host: {0}

Explanation: The MQIPT host cannot be found.

User Response: Check you have correctly specified the hostname where the MQIPT is located.

MQCAE002 The following error was reported by the system: {0}

Explanation: An error has occurred. While following a system command, an error was reported.

MQCAE005 No valid destination address has been defined

Explanation: When adding a route, the destination field was 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 was 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 was left blank.

User Response: Enter a valid listener port address, between 1 and 65535.

MQCAE008 No valid network address has been defined

Explanation: When adding an MQIPT, the network address field was 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 was used.

User Response: Enter a valid command port address, between 1 and 65535.

MQCAE010 Could not show online help

Explanation: The file for online help was available but could not be displayed.

User Response: Make sure you have a web browser 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 Technical 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 Technical Support if the condition persists.

MQCAE015 The password you have just entered has not been recognized

Explanation: The MQIPT expects a valid password, the one used in the last command was incorrect. It must match the one defined in the configuration file.

User Response: Change the password using the **MQIPT->Connection** panel and retry the last command.

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. Contact IBM Technical 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 :

- look in the Readme file for a possible new message
- ″guiadmin.jar″ file is in the system CLASSPATH
- ″guiadmin.properties″ file is in the ″guiadmin.jar″ file
- message number is in the ″guiadmin.properties″ file

MQCAE018 Could not create NLS text for message MQCAE017

Explanation: Message number {0} cannot be found in the system property list.

User Response: The ″guiadmin.properties″ file could be corrupted, check the following:

- ″guiadmin.jar″ file is in the system CLASSPATH
- ″guiadmin.properties″ file is in the ″guiadmin.jar″ file
- 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 retry the command. If the condition persists contact IBM Technical 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 retry the command. If the condition persists contact IBM Technical 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 retry the command. If the condition persists contact IBM Technical Support.

MQCAE023 MQIPT {0} did not recognize your password.

Explanation: The MQIPT expects a valid password, the one used in the last command was incorrect. It must match the one defined in the configuration file

User Response: Change the password using the menu **MQIPT->Connection** panel and retry the command.

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 that 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 does not work, 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 RemoteShutDown property to true.

MQCAE027 Look and feel {0} is not supported.

Explanation: The recommended Look&Feel for the platform you are using is not available.

User Response: Processing continues with the system default Look&Feel.

MQCAE028 Look and feel class {0} cannot be found.

Explanation: The recommended Look&Feel for the platform you are using is not available.

User Response: Processing continues with the system default Look&Feel.

MQCAE029 Minimum Connection Threads 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 Maximum Connection Threads 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: You are attempting to set a value that does not meet the specification.

User Response: Change the value accordingly.

MQCAE033 Max Log file size must be in the range 5 to 50

Explanation: You are attempting to set a value that does not meet the specification.

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.

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:

- MQIPT is not running.
- MQIPT is not listening on its command port.
- Only one Administration Client is using the MQIPT CommandPort.
- 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 that 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: Check that MQIPT has write access to the configuration file.

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:

- MQIPT is not running.
- MQIPT is not listening on its command port.
- Only one Administration Client is using the MQIPT CommandPort.
- 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 appropriate 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 a different listener port and try again.

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 pop-up 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 ″client.conf″ file.

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:

- MQIPT is not running.
- MQIPT is not listening on its command port.
- The request has timed-out. The timeout can be increased by changing the timeout property on the connection information for MQIPT.

MQCAI030 Route {0} is active

Explanation: The MQIPT 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:

- MQIPT is not running.
- MQIPT is not listening on its command port.
- The request has timed-out. The timeout can be increased by changing the timeout property on the connection information for 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.

MQCPE000 Could not locate message data when handling message {0}

Explanation: Message number {0} cannot be found in the system property list.

User Response: The ″mqipt.properties″ file has become corrupted and the specified message number could not be found. Check the following:

- ″MQipt.jar″ file is in the system CLASSPATH
- ″mqipt.properties″ file is in the ″MQipt.jar″ file
- message number is in the ″mqipt.properties″ file

MQCPE001 Directory does not exist or is not a directory

Explanation: At 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 on 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 number supplied for the command port or listener port parameter is invalid.

User Response: Specify a valid port number that is free for use. For guidance on use of port numbers in your network, consult your network administrator.

MQCPE011 The trace level {0} is outside the valid range 0 - 5

Explanation: The specified trace option was requested, but it is not in the valid range 0-5.

User Response: Specify a trace value of 0–5.

MQCPE012 The value {0} is not valid for the attribute {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 control parameter.

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 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 0–65535. 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 has become corrupted and the specified message number could not be found. Check the following:

- look in the Readme file for a possible new message
- ″MQipt.jar″ file is in the system CLASSPATH
- ″mqipt.properties″ file is in the ″MQipt.jar″ file
- 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 has specified the minimum number of connection threads with a value exceeding 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 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 Technical 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] entries.

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 property Destination was not supplied for route {0}

Explanation: The property Destination is mandatory within 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 {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 command 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 Technical Support.

MQCPE041 Unsupported level of Java - {0}

Explanation: MQIPT has been started using the specified level of Java.

User Response: Check the prerequisites in the Users Guide for more information.

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 name is missing

Explanation: The HTTPProxy property must be set if the HTTP property has been set to true.

User Response: Edit the configuration file and define an HTTPProxy for the given route.

MQCPE046 {0} is not allowed as Pagent has failed to initialize

Explanation: Pagent is the application that provides the Quality of Service for MQIPT. MQIPT has failed to initialize it during startup and the QoS property has been set to true for the given route.

User Response: Edit the configuration file and disable QoS for the given route.

MQCPE047 Pagent has failed to initialize

Explanation: Pagent is the application that provides the Quality of Service for MQIPT. MQIPT has failed to initialize it during startup.

User Response: This error message can be ignored if Pagent is not being used but you must set the QoS property to false.

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 {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 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.

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 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.

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 lease 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 LDAPServer{0}Password property

Explanation: An LDAP userid has been specified without a password.

User Response: Edit the configuration file and define an LDAPServer*Password for the given route.

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.

MQCPE069 Invalid port address {0} in security exit response

Explanation: The port address specified in the SecurityExitResponse is not valid.

User Response: The port address must be in the range 1024-65535.

MQCPE070 Unknown reason code {0) in security exit response

Explanation: The reason code specified in the SecurityExitResponse is not supported.

User Response: Refer to the MQIPT book for a list of supported reason codes.

MQCPE071 Error writing to {0}

Explanation: An error occurred while creating or updating the specified file. The error message also contains the exception thrown.

User Response: 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.

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 it is recommended that a second MQIPT instance is created and the routes split between the two.

MQCPI020 The configuration file has been sent to the Administration Client.

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 ....using HTTP proxy {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 ......key ring 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 MQSeries 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 ....via Socks proxy {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: The fork process will sleep for this amount of time before it checks if MQIPT has started successfully as a system service.

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.

MQCPI049 ....QoS priority to dest = {0}, to caller = {1}

Explanation: This shows the priority of traffic in both directions along this route.

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.

MQCPI056 The Java Security Manager must be restarted to use a new policy file

Explanation: The SecurityManagerPolicy property has been changed, but will not come into effect until the Java Security Manager is restarted.

User Response: Change the SecurityManager property to false, issue a refresh command to stop the Java Security Manager. Then change the SecurityManager back to true and issue another refresh command to start the Java Security Manager with the new 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.

MQCPI065 ....in SSL proxy mode

Explanation: This message is written to the system console when a route is started and shows this route is working in SSL proxy mode.

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.

MQCPI067 Setting up links to TQoS runtime libraries

Explanation: User has run the mqiptQoS script to link to the real TQoS runtime libraries.

MQCPI068 Removing links to TQoS runtime libraries

Explanation: User has run the mqiptQoS script to remove links to the real TQoS runtime libraries.

MQCPI069 ....binding to local address {0}

Explanation: This message shows the local IP address each connection is bound to. This should only be used on a multihomed system.

MQCPI070 ....using local port address range {0}-{10}

Explanation: This message shows the local port addresses that will be used for a connection. 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} seconds

Explanation: This message is written to the system console when a route is started. Used to show the timeout value of the security exit.

MQCPI081 Start message for WebSphere MQ internet pass-thru

Explanation: Start message for WebSphere MQ internet pass-thru as a service

MQCPI082 Stop message for WebSphere MQ internet pass-thru

Explanation: Stop message for WebSphere 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.

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.

MQCPI117 This script is used to manage links to the TQoS runtime libraries

Explanation: Online help message from mqiptService script.

MQCPI118 mqiptQoS (-install | -remove)

Explanation: Online help message from mqiptService script.

MQCPI119 -install will setup links to the real TQoS runtime libraries

Explanation: Online help message from mqiptService script.

MQCPI120 -remove will remove links to the real TQoS runtime libraries

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 mqiptQoS script.

MQCPW001 CRL expired for {0}

Explanation: This message is displayed when a CRL (or ARL) is retrieved from an LDAP server or key ring file.

User Response: Update the specified CRL in the LDAP server or key ring file.

MQCPW002 Error updating key ring file {0} with CRL

Explanation: This message is displayed when the LDAPSaveCRLs property has been enabled and the specified key ring file cannot be updated.

User Response: The specified file may have become corrupted. Check the following :

1. write access must be enabled for MQIPT
2. the file is not open by another application

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.

Appendix. Notices

The following paragraph does not apply in any 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 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.

References in this publication to IBM products, programs, or services do not imply that IBM intends to make these available in all countries in which IBM operates.

Any reference to an IBM licensed program or other IBM product in this publication is not intended to state or imply that only IBM's program or other product may be used. Any functionally equivalent program that does not infringe any of the intellectual property rights may be used instead of the IBM product. Evaluation and verification of operation in conjunction with other products, except those expressly designated by IBM, is the user's responsibility.

IBM may have patents or pending patent applications covering subject matter in this document. The furnishing of this document does not give you any license to these patents. You can send license inquiries, in writing, to the IBM Director of Licensing, IBM Corporation, 500 Columbus Avenue, Thornwood, New York 10594, USA.

The information contained in this document has not be submitted to any formal IBM test and is distributed AS IS. The use of the information or the implementation of any of these techniques is a customer responsibility and depends on the customer's ability to evaluate and integrate them into the customer's operational environment. While each item has been reviewed by IBM for accuracy in a specific situation, there is no guarantee that the same or similar results will be obtained elsewhere. Customers attempting to adapt these techniques to their own environments do so at their own risk.

Trademarks

The following terms are trademarks of International Business Machines Corporation in the United States, or other countries, or both:

AIX
FFST
First Failure Support Technology
IBM
IBMLink
MQSeries
SupportPac
WebSphere

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

Java and all Java-based trademarks are trademarks of Sun Microsystems, Inc. in the United States, other countries, or both.

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

Other company, product and service names may be trademarks or service marks of others.

Bibliography

This book is available in HTML as part of the installed product. The HTML is contained in a self extracting zip file in the `doc\<locale>\html\<filename>.zip` directory. Before using the Administration Client you must unzip the file found in the `<locale>\html` subdirectory. The book has been produced in the following languages, see the table below for the language and corresponding filename:

Table 4. Summary of languages and filenames

Language	Locale	HTML filename
Simplified Chinese	zn_CN	amqyzb01.zip
German	de_DE	amqygb01.zip
Japanese	ja_JP	amqyjb01.zip
Korean	ko_KR	amqykb01.zip
Brazilian Portuguese	pt_BR	amqybb01.zip
Spanish	es_ES	amqysb01.zip
US English	en_US	amqyab01.zip

Translated PDFs can be downloaded from the following URL:

`http://www.ibm.com/webspheremq/downloads`

It is available in the following languages:

Table 5. PDF languages and filenames

Language	Locale	PDF filename
Simplified Chinese	zn_CN	amqyzb01.pdf
German	de_DE	amqygb01.pdf
Japanese	ja_JP	amqyjb01.pdf
Korean	ko_KR	amqykb01.pdf
Brazilian Portuguese	pt_BR	amqybb01.pdf
Spanish	es_ES	amqysb01.pdf
US English	en_US	amqyab01.pdf

You will also find the following publications useful:

- *WebSphere MQ Intercommunication*, SC34-6059
- *WebSphere MQ System Administration Guide*, SC34-6068
- *WebSphere MQ Clients*, GC34-6058
- *WebSphere MQ Queue Manager Clusters*, SC34-6061

These books provide information about the definition of WebSphere MQ channels and their attributes - in particular, the definition of CONNAME.

The WebSphere MQ publications are available at:

`http://www.ibm.com/webspheremq/library`

Index

Sending your comments to IBM

If you especially like or dislike anything about this book, please use one of the methods listed below to send your comments to IBM.

Feel free to comment on what you regard as specific errors or omissions, and on the accuracy, organization, subject matter, or completeness of this book.

Please limit your comments to the information in this book and the way in which the information is presented.

To make comments about the functions of IBM products or systems, talk to your IBM representative or to your IBM authorized remarketer.

When you send comments 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.

You can send your comments to IBM in any of the following ways:

- By mail, to this address:

  User Technologies Department (MP095)
  IBM United Kingdom Laboratories
  Hursley Park
  WINCHESTER,
  Hampshire
  SO21 2JN
  United Kingdom
- By fax:
  - From outside the U.K., after your international access code use 44–1962–816151
  - From within the U.K., use 01962–816151
- Electronically, use the appropriate network ID:
  - IBM Mail Exchange: GBIBM2Q9 at IBMMAIL
  - IBMLink™: HURSLEY(IDRCF)
  - Internet: idrcf@hursley.ibm.com

Whichever method you use, ensure that you include:

- The publication title and order number
- The topic to which your comment applies
- Your name and address/telephone number/fax number/network ID.