WebSphere MQ

Clients

Version 7.0

GC34-6934-01

WebSphere MQ

Clients

Version 7.0

GC34-6934-01

Note

Before using this information and the product it supports, be sure to read the general information under notices at the back of this book.

Second edition (January 2009)

This edition of the book applies to the following products:

- IBM WebSphere MQ, Version 7.0
- IBM WebSphere MQ for z/OS, Version 7.0

and to any subsequent releases and modifications until otherwise indicated in new editions.

Contents

Figures

Tables

Chapter 1. Introduction to WebSphere MQ clients

Overview of WebSphere MQ clients

This chapter discusses the following topics:
- "What is a WebSphere MQ client?"
- "Why use WebSphere MQ clients?" on page 4
- "How do I set up a WebSphere MQ client?" on page 5

What is a WebSphere MQ client?

A *WebSphere® MQ client* is a component of the WebSphere MQ product that can be installed on a system on which no queue manager runs.

It enables an application, running on the same system as the WebSphere MQ client, to connect to a queue manager that is running on another system and issue MQI calls to that queue manager. Such an application is called a *WebSphere MQ client application* and the queue manager is referred to as a *server queue manager*.

A WebSphere MQ client application and a server queue manager communicate with each other by using an *MQI channel*. An MQI channel starts when the client application issues an MQCONN or MQCONNX call to connect to the queue manager and ends when the client application issues an MQDISC call to disconnect from the queue manager. The input parameters of an MQI call flow in one direction on an MQI channel and the output parameters flow in the opposite direction.

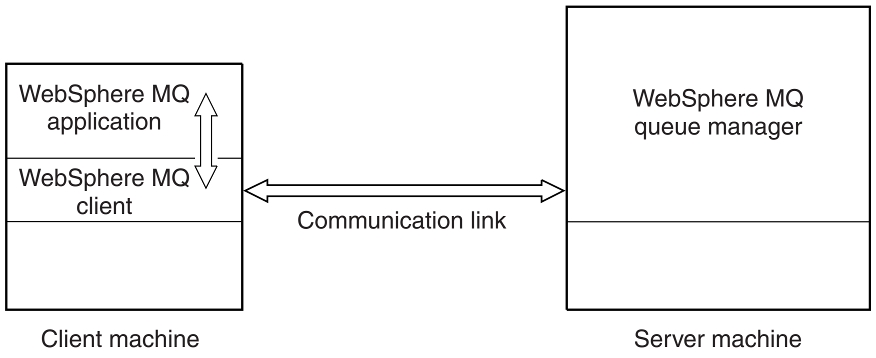

Figure 1. Link between a client and server

The following platforms can be used. The combinations depend on which WebSphere MQ product you are using and are described in "Platform support for WebSphere MQ clients" on page 7.

WebSphere MQ client	WebSphere MQ server
UNIX systems	i5/OS
Windows	UNIX systems
	Windows
	z/OS

The MQI is available to applications running on the client platform; the queues and other WebSphere MQ objects are held on a queue manager that you have installed on a server machine.

An application that you want to run in the WebSphere MQ client environment must first be linked with the relevant client library. When the application issues an MQI call, the WebSphere MQ client directs the request to a queue manager, where it is processed and from where a reply is sent back to the WebSphere MQ client.

The link between the application and the WebSphere MQ client is established dynamically at runtime.

You can also develop client applications using the WebSphere MQ classes for .NET, WebSphere MQ classes for Java™ or WebSphere MQ classes for Java Message Service (JMS). You can use Java and JMS clients on i5/OS® as well as on UNIX® and Windows®. The use of Java and JMS is not described in this book. For full details on how to install, configure, and use WebSphere MQ classes for Java and WebSphere MQ classes for JMS see WebSphere MQ Using Java.

How the client connects to the server

An application running in the WebSphere MQ client environment must maintain an active connection between the client and server machines.

The connection is made by an application issuing an MQCONN or MQCONNX call. Clients and servers communicate through *MQI channels*, or, when using sharing conversations, conversations each share an MQI channel instance. When the call succeeds, the MQI channel instance or conversation remains connected until the application issues a MQDISC call. This is the case for every queue manager that an application needs to connect to.

Client and queue manager on the same machine:

You can also run an application in the WebSphere MQ client environment when your machine also has a queue manager installed.

In this situation, you have the choice of linking to the queue manager libraries or the client libraries, but remember that if you link to the client libraries, you still need to define the channel connections. This can be useful during the development phase of an application. You can test your program on your own machine, with no dependency on others, and be confident that it will still work when you move it to an independent WebSphere MQ client environment.

Clients on different platforms:

Here is another example of a WebSphere MQ client and server system. In this example, the server machine communicates with three WebSphere MQ clients on different platforms.

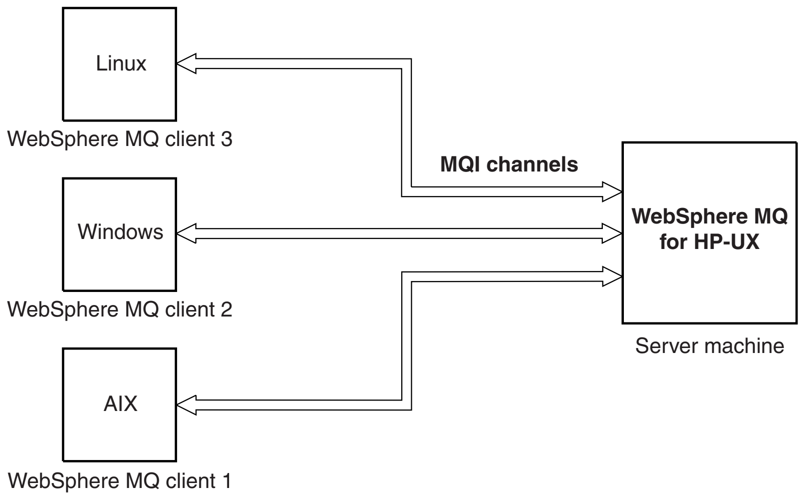

Figure 2. WebSphere MQ server connected to clients on different platforms

Other more complex environments are possible. For example, a WebSphere MQ client can connect to more than one queue manager, or any number of queue managers connected as part of a queue-sharing group.

Using different versions of client and server software:

If you are using previous versions of WebSphere MQ products, make sure that code conversion from the CCSID of your client is supported by the server.

| A WebSphere MQ Version 7.0 client can connect to all supported versions of queue
| manager, whether Version 7.0 or earlier. If you are connecting to an earlier version
| queue manager, you cannot use Version 7.0 features and structures in your
| WebSphere MQ application on the client.

| A WebSphere MQ Version 7.0 queue manager accepts connections from all
| supported versions of client, whether Version 7.0 or earlier.

See the language support tables in WebSphere MQ Application Programming Guide for more information.

What is an extended transactional client?

If you are not familiar with the concepts of transaction management, see Chapter 5, “A review of transaction management,” on page 147.

A client application can participate in a unit of work that is managed by a queue manager to which it is connected. Within the unit of work, the client application can put messages to, and get messages from, the queues that are owned by that queue manager. The client application can then use the MQCMIT call to commit the unit of work or the MQBACK call to back out the unit of work. However, within the same unit of work, the client application cannot update the resources of another resource manager, the tables of a DB2® database, for example. Using a WebSphere MQ extended transactional client removes this restriction.

A *WebSphere MQ extended transactional client* is a WebSphere MQ client with some additional function. This function allows a client application, within the same unit of work:

- To put messages to, and get messages from, queues that are owned by the queue manager to which it is connected
- To update the resources of a resource manager other than a WebSphere MQ queue manager

This unit of work must be managed by an external transaction manager that is running on the same system as the client application. The unit of work cannot be managed by the queue manager to which the client application is connected. This means that the queue manager can act only as a resource manager, not as a transaction manager. It also means that the client application can commit or back out the unit of work using only the application programming interface (API) provided by the external transaction manager. The client application cannot, therefore, use the MQI calls, MQBEGIN, MQCMIT, and MQBACK.

The external transaction manager communicates with the queue manager as a resource manager using the same MQI channel as that used by the client application that is connected to the queue manager. However, in a recovery situation following a failure, when no applications are running, the transaction manager can use a dedicated MQI channel to recover any incomplete units of work in which the queue manager was participating at the time of the failure.

In this book, a WebSphere MQ client that does *not* have the extended transactional function is referred to as a *WebSphere MQ base client*. You can consider, therefore, a WebSphere MQ extended transactional client to consist of a WebSphere MQ base client with the addition of the extended transactional function.

Why use WebSphere MQ clients?

Using WebSphere MQ clients is an efficient way of implementing WebSphere MQ messaging and queuing.

You can have an application that uses the MQI running on one machine and the queue manager running on a different machine (either physical or virtual). The benefits of doing this are:

- There is no need for a full WebSphere MQ implementation on the client machine.
- Hardware requirements on the client system are reduced.
- System administration requirements are reduced.
- a WebSphere MQ application running on a client can connect to multiple queue managers on different systems.
- Alternative channels using different transmission protocols can be used.

What applications run on a WebSphere MQ client?

The full MQI is supported in the client environment.

This enables almost any WebSphere MQ application to be configured to run on a WebSphere MQ client system by linking the application on the WebSphere MQ client to the MQIC library, rather than to the MQI library. The exceptions are:

- MQGET with signal
- An application that needs syncpoint coordination with other resource managers must use an extended transactional client

If read ahead is enabled, to improve non persistent messaging performance, not all MQGET options are available. The table shows the options that are allowed, and whether they can be altered between MQGET calls.

Table 1. MQGET options permitted when read ahead is enabled

	Permitted when read ahead is enabled and can be altered between MQGET calls	**Permitted when read ahead is enabled but cannot be altered between MQGET calls[a]**	**MQGET options that are not permitted when read ahead is enabled[b]**
MQGET MD values	MsgId[c] CorrelId[c]	Encoding CodedCharSetId	
MQGET MQGMO options	MQGMO_WAIT MQGMO_NO_WAIT MQGMO_FAIL_IF_QUIESCING MQGMO_BROWSE_FIRST[d] MQGMO_BROWSE_NEXT[d] MQGMO_BROWSE_MESSAGE _UNDER_CURSOR[d]	MQGMO_SYNCPOINT_IF_PERSISTENT MQGMO_NO_SYNCPOINT MQGMO_ACCEPT_TRUNCATED_MSG MQGMO_CONVERT MQGMO_LOGICAL_ORDER MQGMO_COMPLETE_MSG MQGMO_ALL_MSGS_AVAILABLE MQGMO_ALL_SEGMENTS_AVAILABLE MQGMO_MARK_BROWSE_HANDLE MQGMO_MARK_BROWSE_CO_OP MQGMO_UNMARK_BROWSE_CO_OP MQGMO_UNMARK_BROWSE_HANDLE MQGMO_UNMARKED_BROWSE_MSG MQGMO_PROPERTIES_FORCE_MQRFH2 MQGMO_NO_PROPERTIES MQGMO_PROPERTIES_IN_HANDLE MQGMO_PROPERTIES_COMPATIBILITY	MQGMO_SET_SIGNAL MQGMO_SYNCPOINT MQGMO_MARK_SKIP _BACKOUT MQGMO_MSG_UNDER _CURSOR[d] MQGMO_LOCK MQGMO_UNLOCK
MQGMO values		MsgHandle	

1. If these options are altered between MQGET calls an MQRC_OPTIONS_CHANGED reason code will be returned.
2. If these options are specified on the first MQGET call then read ahead will be disabled. If these options are specified on a subsequent MQGET call a reason code MQRC_OPTIONS_ERROR will be returned.
3. The client applications needs to be aware that if the MsgId and CorrelId values are altered between MQGET calls messages with the previous values may have already been sent to the client and will remain in the client read ahead buffer until consumed (or automatically purged).
4. The first MQGET call determines whether messages are to be browsed or got from a queue when read ahead is enabled. If the application attempts to use a combination of browse and get an MQRC_OPTIONS_CHANGED reason code is returned.
5. MQGMO_MSG_UNDER_CURSOR is not possible with read ahead. Messages can be browsed or got when read ahead is enabled but not a combination of both.

An application running on a WebSphere MQ client can connect to more than one queue manager concurrently, or use a queue manager name with an asterisk (*) on an MQCONN or MQCONNX call (see the examples in "Running applications on WebSphere MQ clients" on page 129).

How do I set up a WebSphere MQ client?

To set up a WebSphere MQ client you need to have a WebSphere MQ server already installed and working on a machine, to which your client will connect. The steps involved in setting up a client are:

1. Check that you have a suitable platform for a WebSphere MQ client and that the hardware and software satisfy the requirements. This is described in "Preparing for installation" on page 7.
2. Decide how you are going to install WebSphere MQ on your client machine, and then follow the instructions for your particular combination of client and server platforms. This is described in "Installing client components" on page 22.
3. Ensure that your communication links are configured and connected. This is described in "Configuring communication links" on page 61.

4. Check that your installation is working correctly. This is described in "Verifying the installation" on page 75.
5. When you have the verified WebSphere MQ client installation, consider whether you need to take any action on security. This is described in "Setting up WebSphere MQ client security" on page 83.
6. Set up the channels between the WebSphere MQ client and server that are required by the WebSphere MQ applications you want to run on the client. This is described in "Using channels" on page 86. There are some additional considerations if you are using SSL. These are described in "The Secure Sockets Layer (SSL) on WebSphere MQ clients" on page 100. You might need to use a WebSphere MQ client configuration file or WebSphere MQ *environment variables* to set up the channels. These are described in "Using WebSphere MQ environment variables" on page 111.
7. WebSphere MQ applications are fully described in the WebSphere MQ Application Programming Guide.
8. There are some differences from a queue manager environment to consider when designing, building and running applications in the WebSphere MQ client environment. For information about these differences, see:
   - "Using the message queue interface (MQI)" on page 119
   - "Building applications for WebSphere MQ clients" on page 125
   - "Running applications on WebSphere MQ clients" on page 129
   - "Solving problems" on page 143

Chapter 2. Installing WebSphere MQ clients

Preparing for installation

This chapter discusses the following topics:
- "Platform support for WebSphere MQ clients"
- "Communications" on page 9
- "Hardware and software requirements" on page 10

Platform support for WebSphere MQ clients

The platform support for WebSphere MQ clients and servers is as follows. Any of the products listed is installed as a *Base product and Server* (*Base product and Client Attachment feature* on WebSphere MQ for z/OS®). These WebSphere MQ products can accept connections from the WebSphere MQ clients on the platforms listed, subject to differences in coded character set identifier (CCSID) and communications protocol.

WebSphere MQ on the following platforms:
- i5/OS
- UNIX systems
- Windows
- z/OS

can accept connections from WebSphere MQ clients on the following platforms:
- UNIX systems
- Windows

Platform support for extended transactional clients

WebSphere MQ extended transactional clients are available for all the platforms that support a base client.

Table 2 lists the supported external transaction managers for each platform.

Table 2. The supported external transaction managers for each extended transactional client platform

Extended transactional client platform	Supported external transaction managers
AIX	IBM WebSphere Application Server (WAS) V5.1 IBM WebSphereApplication Server (WAS) V6.0.2 IBM WebSphere Application Server (WAS) V6.1 IBM TXSeries® V6.0 IBM TXSeries V6.1 BEA Tuxedo V9.1 BEA WebLogic Server 9.1

Table 2. The supported external transaction managers for each extended transactional client platform (continued)

Extended transactional client platform	Supported external transaction managers
HP-UX (PA-RISC platform)	IBM WebSphere Application Server (WAS) V5.1 IBM WebSphereApplication Server (WAS) V6.0.2 IBM WebSphere Application Server (WAS) V6.1 IBM TXSeries V6.0 IBM TXSeries V6.1 BEA Tuxedo V9.1 BEA WebLogic Server 9.1
HP-UX (Itanium® platform)	IBM WebSphere Application Server (WAS) V5.1 IBM WebSphereApplication Server (WAS) V6.0.2 IBM WebSphere Application Server (WAS) V6.1 IBM TXSeries V6.1 BEA Tuxedo V9.1 BEA WebLogic Server 9.1
i5/OS	IBM WebSphere Application Server (WAS) V5.1 IBM WebSphereApplication Server (WAS) V6.0.2 IBM WebSphere Application Server (WAS) V6.1 IBMCICS Transaction Server for i5/OS
Linux (x86 and x86-64 platforms)	IBM WebSphere Application Server (WAS) V5.1 IBM WebSphereApplication Server (WAS) V6.0.2 IBM WebSphere Application Server (WAS) V6.1 IBM TXSeries V6.1 BEA Tuxedo V9.1 BEA WebLogic Server 9.1
Linux (POWER™ platform)	IBM WebSphere Application Server (WAS) V5.1 IBM WebSphereApplication Server (WAS) V6.0.2 IBM WebSphere Application Server (WAS) V6.1 IBM TXSeries V6.1
Linux (zSeries® s390x platform)	IBM WebSphere Application Server (WAS) V5.1 IBM WebSphereApplication Server (WAS) V6.0.2 IBM WebSphere Application Server (WAS) V6.1 IBM TXSeries V6.1 BEA Tuxedo V9.1
Solaris (SPARC platform)	IBM WebSphere Application Server (WAS) V5.1 IBM WebSphereApplication Server (WAS) V6.0.2 IBM WebSphere Application Server (WAS) V6.1 IBM TXSeries V6.0 IBM TXSeries V6.1 BEA Tuxedo V9.1 BEA WebLogic Server 9.1
Solaris (x86-64 platform)	IBM WebSphere Application Server (WAS) V6.1 IBM TXSeries V6.1 BEA WebLogic Server 9.1
Windows systems	IBM WebSphere Application Server (WAS) V5.1 IBM WebSphereApplication Server (WAS) V6.0.2 IBM WebSphere Application Server (WAS) V6.1 IBM TXSeries V6.0 IBM TXSeries V6.1 BEA Tuxedo V9.1 BEA WebLogic Server 9.1 Microsoft transaction server (MTS)/COM+

A client application that is using an extended transactional client can connect to a queue manager of the following WebSphere MQ Version 7.0 products only:

- WebSphere MQ for AIX®
- WebSphere MQ for HP-UX
- WebSphere MQ for i5/OS
- WebSphere MQ for Linux® (x86 platform)
- WebSphere MQ for Linux (zSeries platform)
- WebSphere MQ for Solaris
- WebSphere MQ for Windows
- WebSphere MQ for z/OS

Even though there is no extended transactional client that runs on z/OS, a client application that is using an extended transactional client can still connect to a queue manager that runs on z/OS.

For each platform, the hardware and software requirements for the extended transactional client are the same as those for the WebSphere MQ base client. A programming language is supported by an extended transactional client if it is supported by the WebSphere MQ base client and by the transaction manager you are using.

Communications

WebSphere MQ clients use MQI channels to communicate with the server.

A channel definition must be created at both the WebSphere MQ client and server ends of the connection. How to do this is explained in "Defining MQI channels" on page 89.

The transmission protocols possible are shown in the following table:

Table 3. Transmission protocols for MQI channels

Client platform	LU 6.2	TCP/IP	NetBIOS	SPX
UNIX systems	Yes[1]	Yes		
Windows	Yes	Yes	Yes	Yes

Note:

1. LU6.2 is not supported on Linux (POWER platform), Linux (x86-64 platform), Linux (zSeries s390x platform), or Solaris (x86-64 platform)

Table 12 on page 62 shows the possible combinations of WebSphere MQ client and server platforms, using these transmission protocols.

A WebSphere MQ application on a WebSphere MQ client can use all the MQI calls in the same way as when the queue manager is local. MQCONN or MQCONNX associates the WebSphere MQ application with the selected queue manager, creating a *connection handle*. Other calls using that connection handle are then processed by the connected queue manager. WebSphere MQ client communication requires an active connection between the client and server machine, in contrast to communication between queue managers, which is connection-independent and time-independent.

The transmission protocol is specified via the channel definition and does not affect the application. For example, a Windows application can connect to one queue manager over TCP/IP and to another queue manager over NetBIOS.

Performance considerations

The transmission protocol you use might affect the performance of the WebSphere MQ client and server system. For dial-up support over a slow telephone line, it might be advisable to use WebSphere MQ channel compression.

Hardware and software requirements

The following table shows where you can find hardware and software requirements for the client platforms.

Client platform	Page
AIX	"AIX client: hardware and software required"
HP-UX	"HP-UX client: hardware and software required" on page 11
Linux	"Linux client: hardware and software required" on page 14
Solaris	"Solaris client: hardware and software required" on page 18
Windows	"Windows client: hardware and software required" on page 20

For your server platform hardware and software requirements, see the manual that describes installation for your platform.

AIX client: hardware and software required

This topic outlines the hardware and software requirements for the WebSphere MQ client for AIX.

Hardware

| WebSphere MQ for AIX, Version 7.0 runs on any machine that supports the AIX
| operating systems listed in the following section, whether from IBM® or other vendors.

Operating System

The operating systems supported by WebSphere MQ for AIX, Version 7.0 are:

- | AIX V5.3 plus TL04 and appropriate firmware
- | AIX V6.1

| Use the oslevel -r command to determine the level of the operating system you are
| running, including the maintenance level.

Connectivity Requirements

Check that the system has 64-bit compatible communications hardware that supports at least one of the following:

- TCP/IP (IPv4 and IPv6, provided by the operating system)
- IBM Communications Server for AIX V6.3 (SNA)

Optional software:

This software is not required to enable you to run WebSphere MQ, but you might need it to make use of certain features of WebSphere MQ.

Secure Sockets Layer (SSL)

If you want to use the SSL support, you need IBM Global Security Kit V7. This is supplied with WebSphere MQ as one of the components available for installation.

You must also have installed version 7.0.4.11 (or later) of the C++ runtime to use the SSL support.

Compilers

The following compilers are supported for WebSphere MQ for AIX applications:

- IBM C for AIX, V6.0
- IBM XL C Enterprise Edition for AIX V7.0
- IBM XL C Enterprise Edition for AIX V8.0
- IBM VisualAge® C++ Professional for AIX V6.0
- IBM XL C/C++ Enterprise Edition for AIX V7.0
- IBM XL C/C++ Enterprise Edition for AIX V8.0

Figure 3. C/C++

- IBM COBOL Set for AIX V2.0 (32-bit applications only)
- Micro Focus Server Express V4.0
- Micro Focus Server Express V5.0

Figure 4. Cobol

- 32-bit
  - IBM 32-bit SDK V1.4.2, 32-bit version
  - IBM 32-bit SDK V5 (for AIX 5.3.0.30 or later)
  - IBM 32-bit SDK V6 (for AIX 5.3.0.30 or later)
- 64-bit
  - IBM 64-bit SDK for AIX Java 2 Technology Edition V1.4.2,
  - IBM 64-bit SDK for AIX Java V5 (for AIX 5.3.0.30 or later)
  - IBM 64-bit SDK for AIX Java V6 (for AIX 5.3.0.30 or later)

Figure 5. Java

HP-UX client: hardware and software required

This topic outlines the hardware and software requirements for the WebSphere MQ client for HP-UX.

Hardware

The WebSphere MQ client for HP-UX runs on any Hewlett Packard PA-RISC 64-bit computer or any Itanium system hardware that is explicitly compatible and fully capable of running the specified operating system, all the corresponding supporting software shown below, and any associated applications unmodified.

Operating System (PA-RISC)

The operating systems supported by WebSphere MQ for HP-UX, Version 7.0, on the PA-RISC platform are:
- HP-UX 11i v2 (11.23)
- HP-UX 11i v3 (11.31)

Operating System (Itanium)

The operating systems supported by WebSphere MQ for HP-UX, Version 7.0, on the Itanium platform are:
- HP-UX 11i v2 (11.23) for IPF
- HP-UX 11i v3 (11.31) for IPF

Connectivity Requirements

Check that the system has 64-bit compatible communications hardware that supports at least one of the following:
- TCP/IP
- HP SNAplus2 Version 7 (For SNA LU 6.2 connectivity)

TCP/IP is part of the base operating system.

IPv6 feature support is available with HP Transport Optional Upgrade Release (TOUR)

Optional software:

This software is not required to enable you to run WebSphere MQ, but you might need it to make use of certain features of WebSphere MQ.

Secure Sockets Layer (SSL)

If you want to use the SSL support with WebSphere MQ for HP-UX, you need to install the IBM Global Security Kit (GSKit) V7 packages: gsk7bas64 and (for the PA-RISC platform only) gsk7bas. These are supplied with WebSphere MQ as components available for installation. On the PA-RISC platform, if you are migrating from WebSphere MQ Version 5.3 and have no other requirement for the IBM Global Security Kit V6 you can uninstall it using the process described in the topic *Uninstalling WebSphere MQ* in *WebSphere MQ for HP-UX Quick Beginnings.*, the package name is `gsk6bas`.

To use SSL, WebSphere MQ clients on HP-UX 11i v1 (PA-RISC only) and HP-UX 11i v2 must be built:
- Using the C++ compiler (not the C compiler)
- Using POSIX threads
- With the compiler options: `-W1,+b/opt/ibm/gsk7/lib:/opt/mqm/lib`

Compilers for HP-UX (PA-RISC platform) applications

The following compilers are supported for WebSphere MQ for HP-UX applications on the PA-RISC platform:

- HP C/ANSI C Developer Bundle for HP-UX 11i2
- HP aCC Version A.06.12

Figure 6. C/C++

- Micro Focus Server Express V4.0
- Micro Focus Server Express V5.0

Figure 7. COBOL

- 32 bit
  - HP-UX SDK for the Java platform and JDK V1.4
  - HP SDK for J2SE HP-UX 11i platform, adapted by IBM for IBM software V1.4.2
  - HP SDK for J2SE HP-UX 11i platform, adapted by IBM for IBM software V5.0 (SR1 or above)
  - HP SDK for J2SE HP-UX 11i platform, adapted by IBM for IBM software V6.0
- 64 bit:
  - HP-UX SDK for the Java platform and JDK V1.4
  - IBM 64-bit SDK for HP-UX Java 2 Technology Edition V1.4.2
  - IBM 64-bit SDK for HP-UX Java 2 Technology Edition V5.0 (SR1 or above)
  - IBM 64-bit SDK for HP-UX Java 2 Technology Edition V6.0 (SR1 or above)

Figure 8. Java

Compilers for HP-UX (Itanium platform) applications

The following compilers are supported for WebSphere MQ for HP-UX applications on the Itanium platform:

- HP C/ANSI C Developer's Bundle V6.02 or later maintenance
- HP aCC A.06.12

Figure 9. C/C++

- Micro Focus Server Express V4.0
- Micro Focus Server Express V5.0

Figure 10. COBOL

- 32-bit
  - HP-UX IPF Software Development Kit for the Java™ 2 platform
  - HP SDK for J2SE HP-UX 11i platform, adapted by IBM® for IBM Software, V1.4.2 for 32-bit Itanium
  - HP SDK for J2SE HP-UX 11i platform, adapted by IBM for IBM Software, V5.0 for 32-bit Itanium (SR1 or above)
  - HP SDK for J2SE HP-UX 11i platform, adapted by IBM for IBM Software, V6.0 for 32-bit Itanium
- 64-bit
  - HP-UX IPF Software Development Kit for the Java 2 platform
  - HP SDK for J2SE HP-UX 11i platform, adapted by IBM for IBM Software, V1.4.2 for 64-bit Itanium
  - HP SDK for J2SE HP-UX 11i platform, adapted by IBM for IBM Software, V5.0 for 64-bit Itanium (SR1 or above)
  - HP SDK for J2SE HP-UX 11i platform, adapted by IBM for IBM Software, V6.0 for 64-bit Itanium

Figure 11. Java

Linux client: hardware and software required

This topic outlines the hardware and software requirements for the WebSphere MQ client for Linux.

Hardware

WebSphere MQ for Linux, Version 7.0 (x86 platform) runs on any computer that supports the x86 machine architecture, capable of running the required level of a compatible operating system.

WebSphere MQ for Linux, Version 7.0 (x86-64 platform) runs on any computer that supports the x86-64 (AMD64, Intel® EM64T or compatible) machine architecture, capable of running the required level of a compatible operating system.

WebSphere MQ for Linux, Version 7.0 (POWER platform) runs on any 64-bit System i or System p IBM POWER processor-based systems only, capable of running the required level of a compatible operating system.

WebSphere MQ for Linux, Version 7.0 (zSeries s390x platform) runs on any IBM System z9 or IBM eServer (or equivalent) 64-bit processor that is explicitly compatible and fully capable of running a specified operating system.

Connectivity Requirements

The network protocols supported by WebSphere MQ for Linux, Version 7.0 (x86 platform) are:

Protocol	Comments
TCP/IP	TCP/IP is part of the Linux (x86 platform) operating system. You can use any communications hardware supporting TCP/IP
LU6.2	If you want to use the SNA LU6.2 support on WebSphere MQ for Linux, Version 7.0 (x86 platform) you need the IBM Communications Server for Linux Version 6.2. The Communications Server is available as a PRPQ product from IBM. For more details, see: http://www.ibm.com/software/network/commserver/about

The network protocols supported by WebSphere MQ for Linux, Version 7.0 (x86-64 platform) are:

Protocol	Comments
TCP/IP	TCP/IP is part of the Linux (x86-64 platform) operating system. You can use any communications hardware supporting TCP/IP

The network protocols supported by WebSphere MQ for Linux, Version 7.0 (zSeries s390x platform) are:

Protocol	Comments
TCP/IP	TCP/IP is part of the Linux (zSeries s390x platform) operating system. You can use any communications hardware supporting TCP/IP

The network protocols supported by WebSphere MQ for Linux, Version 7.0 (POWER platform) are:

Protocol	Comments
TCP/IP	TCP/IP is part of the Linux (POWER platform) operating system. You can use any communications hardware supporting TCP/IP
LU6.2	If you want to use the SNA LU6.2 support on WebSphere MQ for Linux, Version 7.0 (POWER platform) you need the IBM Communications Server for Linux Version 6.2. The Communications Server is available as a PRPQ product from IBM. For more details, see: http://www.ibm.com/software/network/commserver/about

Operating System, WebSphere MQ for Linux, Version 7.0 (x86 platform)

WebSphere MQ for Linux, Version 7.0 (x86 platform) has been tested with the following distributions:

| • Red Hat Enterprise Linux (RHEL) V4.0 or later update
| • Red Hat Enterprise Linux (RHEL) V5.0
| • SuSE Linux Enterprise Server (SLES) V9
| • SuSE Linux Enterprise Server (SLES) V10
| • NLPOS9 FP1
| • IRES V2
| • Red Flag Data Centre V5.0

Operating System, WebSphere MQ for Linux, Version 7.0 (x86-64 platform)

WebSphere MQ for Linux, Version 7.0 (x86-64 platform) has been tested with the following distributions:

| • Red Hat Enterprise Linux V4.0 or later update
| • Red Hat Enterprise Linux (RHEL) V5.0
• SuSE Linux Enterprise Server (SLES) 9
| • SuSE Linux Enterprise Server (SLES) 10

Operating System, WebSphere MQ for Linux, Version 7.0 (POWER platform)

WebSphere MQ for Linux, Version 7.0 (POWER platform) has been tested with the following distributions:

- Red Hat Enterprise Linux (RHEL) V4.0
- Red Hat Enterprise Linux (RHEL) V5.0
- SuSE Linux Enterprise Server (SLES) V9
- SuSE Linux Enterprise Server (SLES) V10

Operating System, WebSphere MQ for Linux, Version 7.0 (zSeries s390x platform)

WebSphere MQ for Linux, Version 7.0 (zSeries s390x platform) has been tested with the following distributions:

- Red Hat Enterprise Linux V4.0
- Red Hat Enterprise Linux (RHEL) V5.0
- SuSE Linux Enterprise Server (SLES) 9
- SuSE Linux Enterprise Server (SLES) V10

Optional software for Linux client:

This software is not required to enable you to run WebSphere MQ, but you might need it to make use of certain features of WebSphere MQ.

Compilers for WebSphere MQ for Linux (x86 platform) applications

The following compilers are supported for WebSphere MQ for Linux, Version 7.0.

C applications/C++ applications

- GNU C Compiler (gcc) and g++ Version 3.3 (SLES/9)
- GNU C Compiler (gcc) and g++ Version 4.1.1 (SLES/10 & RedHat/5)
- GNU C Compiler (gcc) and g++ Version 3.4 (RedHat/4)

Note: The C++ support libraries are installed in directories whose names match the compiler version, `/opt/mqm/lib/<version>`, and links are placed from `/opt/mqm/lib` to the default version, 3.3.

COBOL applications

- Micro Focus Server Express, V4.0
- Micro Focus Server Express, V5.0

Java applications

- IBM 32-bit SDK for Linux on Intel, Java 2 Technology Edition V1.4.2
- IBM 32-bit SDK for Linux on Intel architecture, Java 2 Technology Edition V5.0
- IBM 32-bit SDK for Linux on Intel architecture, Java 2 Technology Edition V6.0
- Java 2 Platform, Standard Edition (J2SE) V1.4.2 from Sun Microsystems, Inc.

Compilers for WebSphere MQ for Linux (x86-64 platform) applications

The following compilers are supported for WebSphere MQ for Linux, Version 7.0.

C applications/C++ applications
- GNU C Compiler (gcc) and g++, Version 3.3 (SLES/9)
- GNU C Compiler (gcc) and g++, Version 4.1.1 (SLES/10 & RedHat/5)
- GNU C Compiler (gcc) and g++, Version 3.4 (RedHat/4)

Note: The C++ support libraries are installed in directories whose names match the compiler version, `/opt/mqm/lib/<version>`, and links are placed from `/opt/mqm/lib` to the default version, 3.3.

COBOL applications
- Micro Focus Server Express, V4.0
- Micro Focus Server Express, V5.0

Java applications
- 32-bit
  - IBM® 32-bit SDK for Linux on Intel® architecture, Java™ 2 Technology Edition, Version 1.4.2
  - IBM 32-bit SDK for Linux on Intel architecture, Java 2 Technology Edition V5.0 (SR1 or above)
  - IBM 32-bit SDK for Linux on Intel architecture, Java 2 Technology Edition V6.0
  - Java 2 Platform, Standard Edition (J2SE) V1.4.2 from Sun Microsystems, Inc.
  - Java 2 Platform, Standard Edition (J2SE) V1.5 from Sun Microsystems, Inc.
- 64-bit
  - IBM SDK for Linux on AMD64/EM64T architecture, Java 2 Technology Edition, Version 1.4.2
  - IBM SDK for Linux on AMD64/EM64T architecture, Java 2 Technology Edition, Version 5
  - IBM SDK for Linux on AMD64/EM64T architecture, Java 2 Technology Edition, Version 6
  - Java 2 Platform, Standard Edition (J2SE) V1.5 from Sun Microsystems, Inc.

Compilers for WebSphere MQ for Linux (POWER platform) applications

The following compilers are supported for WebSphere MQ for Linux applications on the POWER platform:

C/C++
- GNU C Compiler (gcc) and g++ Version 3.3 (SLES/9)
- GNU C Compiler (gcc) and g++ Version 4.1,1 (SLES/10 & RedHat/5)
- GNU C Compiler (gcc) and g++ Version 3.4 (RedHat/4)

Note: The C++ support libraries are installed in directories whose names match the compiler version, `/opt/mqm/lib/<version>`, and links are placed from `/opt/mqm/lib` to the default version, 3.3.

COBOL

- Micro Focus Server Express, V4.0
- Micro Focus Server Express, V5.0

Java
- 32-bit
  - IBM 32-bit SDK for Linux for System i and System p, Java 2 Technology Edition V1.4.2 (supported on System p only)
  - IBM 32-bit SDK for Linux for IBM System i and System p, Java 2 Technology Edition V5.0 (SR1 or above)
  - IBM 32-bit SDK for Linux for IBM System i and System p, Java 2 Technology Edition V6.0
- 64-bit
  - IBM 64-bit SDK for Linux for IBM System i and System p, Java 2 Technology Edition V1.4.2 (supported on System p only)
  - IBM 64-bit SDK for Linux for IBM System i and System p, Java 2 Technology Edition V5.0 (SR1 or above)
  - IBM 64-bit SDK for Linux for IBM System i and System p, Java 2 Technology Edition V6.0

Compilers for WebSphere MQ for Linux (zSeries s390x platform) applications

The following compilers are supported for WebSphere MQ for Linux applications on the zSeries s390x platform.

C/C++
- GNU C Compiler (gcc) and g++ Version 3.3 (SLES/9)
- GNU C Compiler (gcc) and g++ Version 4.1.1 (SLES/10 & RedHat/5)
- GNU C Compiler (gcc) and g++ Version 3.4 (RedHat/4)

Note: The C++ support libraries are installed in directories whose names match the compiler version, `/opt/mqm/lib/<version>`, and links are placed from `/opt/mqm/lib` to the default version, 3.3.

COBOL
- Micro Focus Server Express, V4.0
- Micro Focus Server Express, V5.0

Java
- IBM 31-bit SDK for Linux on System z9, Java 2 Technology Edition, Version 1.4.2
- IBM 31-bit SDK for Linux on System z9, Java 2 Technology Edition V5.0 (SR1 or above)
- IBM 31-bit SDK for Linux on System z9, Java 2 Technology Edition V6.0
- IBM 64-bit SDK for Linux on System z9, Java 2 Technology Edition, Version 1.4.2
- IBM 64-bit SDK for Linux on System z9, Java 2 Technology Edition V5.0 (SR1 or above)
- IBM 64-bit SDK for Linux on System z9, Java 2 Technology Edition V6.0

Solaris client: hardware and software required

This topic outlines the hardware and software requirements for the WebSphere MQ client for Solaris.

Harware

The WebSphere MQ client for Solaris runs on any64-bit Sun SPARC systems capable of running the required level of a compatible operating system.

Operating System

The operating systems supported by WebSphere MQ for Sun Solaris, Version 7.0 are:

- Sun Solaris 9
- Sun Solaris 10

Connectivity Requirements

Check that the system has 64-bit compatible communications hardware that supports at least one of the following:

- TCP/IP (IPv4 and IPv6, provided by the operating system)
- SNAP-IX Version 7.0 (SNA)

Optional software:

This software is not required to enable you to run WebSphere MQ, but you might need it to make use of certain features of WebSphere MQ.

Secure Sockets Layer (SSL)

If you want to use the SSL support, you need IBM Global Security Kit V7. This is supplied with WebSphere MQ as one of the components available for installation.

Compilers

The following compilers are supported for WebSphere MQ for Sun Solaris applications:

- Sun ONE Studio 9, Compiler Collection
- Sun Studio 10 Software for Solaris Platforms
- Sun ONE Studio 11 Enterprise Edition for Solaris

Figure 12. C/C++

- Micro Focus Server Express V4.0
- Micro Focus Server Express V5.0

Figure 13. COBOL

- 32-bit
  - Sun Solaris Java SDK V1.4 with JDK V1.4.2
  - Sun Solaris Java SDK V5 with JDK V5
  - Sun Solaris Java SDK V6 with JDK V6
  - IBM 32-bit SDK for Solaris, Java 2 Technology Edition V1.4.2
  - IBM 32-bit SDK for Solaris, Java 2 Technology Edition V5.0 (SR1 or above)
  - IBM 32-bit SDK for Solaris, Java 2 Technology Edition V6.0
- 64-bit
  - Sun Java 2 SDK, Standard Edition V1.4.2
  - Sun Java 2 SDK, Standard Edition V5
  - Sun Java 2 SDK, Standard Edition V6
  - IBM® 64-bit SDK for Solaris, Java 2 Technology Edition, Version 1.4.2
  - IBM 64-bit SDK for Solaris, Java 2 Technology Edition V5.0 (SR1 or above)
  - IBM 64-bit SDK for Solaris, Java 2 Technology Edition V6.0

Figure 14. Java

Windows client: hardware and software required

This topic outlines the hardware and software requirements for the WebSphere MQ client for Windows.

Hardware

Any x86 or x86-64 technology-compatible PC hardware, capable of running the required level of a compatible operating system.

Operating system

WebSphere® MQ requires one of the following operating systems:

- Microsoft® Windows Server 2003. This can be one of the following products:
  - Microsoft Windows Server 2003 Standard Edition (Service Pack 1 or later)
  - Microsoft Windows Server 2003 Enterprise Edition (Service Pack 1 or later)
  - Microsoft Windows Server 2003 Standard x64 Edition (Service Pack 1 or later)
  - Microsoft Windows Server 2003 Enterprise x64 Edition (Service Pack 1 or later)
  - Microsoft Windows Server 2003 R2 Standard Edition (Service Pack 1 or later)
  - Microsoft Windows Server 2003 R2 Enterprise Edition (Service Pack 1 or later)
  - Microsoft Windows Server 2003 R2 Standard x64 Edition (Service Pack 1 or later)
  - Microsoft Windows Server 2003 R2 Enterprise x64 Edition (Service Pack 1 or later)
- Microsoft Windows XP Professional. This can be either of the following products:
  - Microsoft Windows XP Professional (Service Pack 2 or later)
  - Microsoft Windows XP Professional x64 Edition
- Microsoft Windows Vista. This can be one of the following products:
  - Microsoft Windows Vista Business Edition
  - Microsoft Windows Vista Enterprise Edition

  - Microsoft Windows Vista Ultimate Edition
  - Microsoft Windows Vista Business x64 Edition
  - Microsoft Windows Vista Enterprise x64 Edition
  - Microsoft Windows Vista Ultimate x64 Edition
- Microsoft Windows Server 2008. This can be one of the following products:
  - Microsoft Windows Server 2008 Standard Edition
  - Microsoft Windows Server 2008 Enterprise Edition
  - Microsoft Windows Server 2008 Datacenter Edition
  - Microsoft Windows Server HPC 2008
  - Microsoft Windows Server Web 2008
  - Microsoft Windows Server Storage 2008
  - Microsoft Windows Server 2008 Standard x64 Edition
  - Microsoft Windows Server 2008 Enterprise x64 Edition
  - Microsoft Windows Server 2008 Datacenter x64 Edition
  - Microsoft Windows Server Web 2008 x64
  - Microsoft Windows Server Storage 2008 x64
  - Microsoft Windows Small Business Server 2008 x64 for small businesses
  - Microsoft Windows Essential Business Server 2008 x64 for medium-sized businesses
  - Microsoft Windows Server 2008 for Itanium-based Systems

Connectivity

You require one of the following products:
- for SNA connectivity:
  - IBM Communications Server for Windows, Version 6.1.2
  - IBM Personal Communications for Windows Version 5.9, part of IBM Host Access Client Package (HACP) V4.0
  - Attachmate myEXTRA! Presentation Services, Version 7.11
  - Attachmate EXTRA! X-treme V9
  - Microsoft Host Integration Server 2006
- TCP/IP, NetBIOS, and SPX. These are part of the base operating system (SPX is part of Windows XP and Windows 2003 only).
- WebSphere MQ client applications are supported on the Citrix Presentation Server V4.5

Optional software:

This software is not required to enable you to run WebSphere MQ, but you might need it to make use of certain features of WebSphere MQ.

Compilers for WebSphere MQ client applications on a Windows system

The following software compilers are supported:
- C and C++:
  - Microsoft Visual Studio C++ 2005 SP1
  - Microsoft Visual Studio C++ .NET 2003

Note: If you compile your application using Microsoft Visual Studio on one system, and then copy the application to another system that does not have Microsoft Visual Studio installed, you must install Microsoft Visual Studio redistributable package (vcredist) on the target system.

- .NET
  - Microsoft Visual C++ .NET 2003
  - Microsoft Visual C++ .NET 2005
  - Microsoft Visual C# .NET 2003
  - Microsoft Visual C# .NET 2005
  - Microsoft Visual Basic .NET 2003
  - Microsoft Visual Basic .NET 2005
- COBOL:
  - IBM VisualAge COBOL Enterprise V3.0.1
  - Micro Focus Net Express Version 4.0
  - Micro Focus Net Express Version 5.0
- Visual Basic:
  - Microsoft Visual Basic, Version 6.0
- JDK:
  - 32-bit
    - IBM Developer Kit for Windows, Java 2 Technology Edition V1.4.2.
    - IBM Developer Kit for Windows, Java 2 Technology Edition V5.0 (SR1 or above)
    - IBM Developer Kit for Windows, Java 2 Technology Edition
    - Java 2 Platform, Standard Edition V1.4.2 from Sun Microsystems, Inc.
    - Java 2 Platform, Standard Edition V5.0 (SR1 or above) from Sun Microsystems, Inc.
    - Java 2 Platform, Standard Edition V6.0 from Sun Microsystems, Inc.
  - 64-bit
    - IBM 64-bit SDK for Windows AMD64/EM64T architecture, Java 2 Technology Edition, Version 1.4.2
    - IBM 64-bit SDK for Windows AMD64/EM64T architecture, Java 2 Technology Edition, Version 5.0
    - IBM 64-bit SDK for Windows AMD64/EM64T architecture, Java 2 Technology Edition, Version 6.0
    - Sun Java 2 Platform Standard Edition, Version 5.0

Other

Microsoft Windows Terminal Server feature.

Installing client components

This chapter discusses how to install the client components for WebSphere MQ Version 7.0 on UNIX systems and on Windows. For UNIX platforms, the main installation instructions can be found in the *Quick beginnings* guide for each platform, while this book provides additional information on the extended transactional client, SSL, migration and so on.

Installing a WebSphere MQ client and server system

The following CDs are supplied with WebSphere MQ:

WebSphere MQ Server CD for each server platform
: Each CD contains the components that can be installed on a server machine. These components include the WebSphere MQ client for the same platform as the server.

WebSphere MQ Client CDs
: A set of CDs containing the base client components that can be installed on a client machine for each of the following platforms:

CD number	Platform
1	AIX
2	HP-UX
3	Linux (x86 platform) Linux (POWER platform) Linux (zSeries platform)
4	Solaris
5	Windows

For Solaris and HP-UX, the CD contains, in two separate directories, the following sets of client components:

- The client components without the WebSphere MQ SSL support
- The client components with the WebSphere MQ SSL support

Only one set of client components is supplied for Windows, and this set contains all the client components that are needed to use the Windows client with or without the WebSphere MQ SSL support.

For AIX and Linux, you can choose whether to install SSL components at installation time.

WebSphere MQ extended transactional clients CD
: A CD containing the extended transactional function for clients on all supported platforms.

To install WebSphere MQ on a server machine, use the Server CD for your platform. For installation instructions, see the *Quick Beginnings* book for your platform.

To install WebSphere MQ on a client machine, use the Client CD that contains the client components for your platform. For installation instructions, see:

- “Installing the WebSphere MQ client on AIX” on page 25
- “Installing on HP-UX” on page 29
- “Installing on Linux” on page 32
- “Installing on Solaris” on page 35
- “Installing on Windows” on page 38

To install the WebSphere MQ extended transactional client on a platform, you must first install the WebSphere MQ base client. You can then install the extended transactional function by using the WebSphere MQ Extended Transactional Clients

CD-ROM. If you want to install the extended transactional function from an installation image that you have downloaded, replace references to the CD-ROM in this chapter by references to the directory containing the installation image.

After you have installed an extended transactional client on a UNIX system, you must use the **setmqcap** control command to declare the number of processors for which you have purchased license units. For information about how to use the **setmqcap** command, see the WebSphere MQ System Administration Guide. On a Windows system, you do not need to issue the **setmqcap** command because you are asked during the installation of the extended transactional function whether you have purchased sufficient license units.

It is possible to install a WebSphere MQ client on the same machine as a WebSphere MQ server by using the appropriate Server CD, as explained in "Installing WebSphere MQ clients on the same machine as the server."

Installing from an electronic software download

It is possible to install the UNIX WebSphere MQ Clients from an installation image downloaded from the IBM download site.

The installation image is provided as a compressed tape archive (tar) file. Follow these steps to install from the tar file:

1. Copy the WebSphere MQ tar file to a suitable directory accessible to the machines where the software is to be installed. This directory must be on a file system with at least the amount of free space indicated below (this is in addition to the disk space required for the product, as detailed in the appropriate section of "Hardware and software requirements" on page 10).

MQ70Client_solaris.tar	120MB
MQ70ClientSSL_solaris.tar	20MB
MQ70Client_hpux.tar	70MB
MQ70ClientSSL_hpux.tar	190MB
MQ70Client_aix.tar	40MB
MQ70ClientSSL_aix.tar	110MB
MQ70Client_LinuxIntel.tar	250MB
MQ70ClientSSL_LinuxIntel.tar	310MB
MQ70Client_LinuxzSeries.tar	85MB
MQ70ClientSSL_LinuxzSeries.tar	140MB

2. Make this directory the current directory and use the command: `tar -xvf .tar` to create the installation image.

   When the command completes, you can delete the tar file
3. Follow the instructions given in the appropriate section of this chapter to install and configure the product. Replace any references to the CD drive by the directory used in the steps above. All other instructions remain the same.

Installing WebSphere MQ clients on the same machine as the server

To install a WebSphere MQ client on a WebSphere MQ server machine, use the appropriate Server CD. Use a Client CD only to install a WebSphere MQ client on a machine that is not a WebSphere MQ server.

You might install a WebSphere MQ client from a Client CD and later decide to install the WebSphere MQ server on the same machine. Before you can do this, you must remove all the client components from the machine. Then use the appropriate Server CD to install the server and client components. You cannot install the WebSphere MQ server on a machine that already has client components installed from a Client CD.

Remember that, even if your client and server reside on the same machine, you still need to define the MQI channel between them. See “Using channels” on page 86 for details.

Uninstalling WebSphere MQ clients

If you want to remove the WebSphere MQ client files from your system, use the process provided to do this efficiently. Details are given after the installation instructions for each platform.

Installing the WebSphere MQ client on AIX

Depending on your situation, you install the WebSphere MQ client in slightly different ways.

To install the WebSphere MQ client on an AIX system, use WebSphere MQ Client CD 1, which is supplied with WebSphere MQ, and follow the instructions in WebSphere MQ for AIX Quick Beginnings.

If you have an earlier version than WebSphere MQ Version 7.0 for AIX client installed on your system, or if a file system remains from a previous AIX client installation, see “Migrating from an earlier version of WebSphere MQ for AIX client” on page 27.

If you plan to install a WebSphere MQ client and server on the same machine, see “Installing WebSphere MQ clients on the same machine as the server” on page 24.

To install the extended transactional client on AIX, you must first install the WebSphere MQ base client. As a minimum, you must install the runtime, client, and client for Java components of the WebSphere MQ base client.

Components for AIX

The components you can install on AIX systems are:

WebSphere MQ client
The WebSphere MQ client code for your UNIX platform.

Samples
Sample application programs.

Runtime component
Support for external applications. This does **not** enable you to write your own applications.

Base
Support to enable you to create and support your own applications. Requires the runtime component to be installed.

WebSphere MQ Client for Java
This allows Java applets running on your client machine to communicate with WebSphere MQ. It includes security exits for encryption and authentication of messages sent across the Web by the WebSphere MQ Client for Java. These exits consist of some Java classes. To use the client for Java you need to have Java runtime code on your machine, at the

following (or later compatible) levels:

- 32-bit
  - IBM 32-bit SDK V1.4.2, 32-bit version
  - IBM 32-bit SDK V5 (for AIX 5.3.0.30 or later)
  - IBM 32-bit SDK V6 (for AIX 5.3.0.30 or later)
- 64-bit
  - IBM 64-bit SDK for AIX Java 2 Technology Edition V1.4.2,
  - IBM 64-bit SDK for AIX Java V5 (for AIX 5.3.0.30 or later)
  - IBM 64-bit SDK for AIX Java V6 (for AIX 5.3.0.30 or later)

Figure 15. Java

For information about Java runtime see the WebSphere MQ Using Java book.

Installing without SSL support

Note: If you have a previous version of the WebSphere MQ for AIX client installed on your system, or if a file system remains from a previous AIX client installation, see “Migrating from an earlier version of WebSphere MQ for AIX client” on page 27.

Follow the installation instructions but choose **not** to install the following filesets:

- mqm.keyman.rte
- gsksa.rte
- gskta.rte

Installing the extended transactional function

This section describes how to install the extended transactional client using the graphical user interface of the System Management Interface Tool (SMIT). You can also install the extended transactional client by using the SMIT terminal interface or the **installp** command.

Before you attempt to install the extended transactional function, end all WebSphere MQ activity on your system by stopping any of the following that might be running:

- WebSphere MQ applications
- Queue managers and their associated listeners

To install the extended transactional function, follow this procedure:

1. Log in as root.
2. Insert the WebSphere MQ Extended Transactional Clients CD-ROM into the CD-ROM drive.
3. From the shell, type `smit` and press Enter.
4. Click the following menu items in sequence:
   a. **Software Installation and Maintenance**
   b. **Install and Update Software**
   c. **Install and Update from ALL Available Software**

   The Install and Update from ALL Available Software window opens.

5. In the **INPUT device / directory for software** field, type the device name of the CD-ROM drive (`/dev/cd0`, for example), or click **List** and select the CD-ROM drive from the list of device names that is displayed.

   Click **OK**. The main Install and Update from ALL Available Software window opens.
6. Click **List** in the **SOFTWARE to install** field, select the mqm.txclient.rte file set, which is the component containing the extended transactional function, and click **OK**.
7. Review the values of the other fields in the window to make sure they are what you require.
8. Click **OK** to install the component you have selected.
9. Click **Done** when the installation is complete.

Migrating from an earlier version of WebSphere MQ for AIX client

If you want to migrate to a WebSphere MQ for AIX, Version 7.0 client from an earlier version of the WebSphere MQ client, you must perform certain tasks.

You must first end all WebSphere MQ activity on the target machine.

This migration procedure applies only to migration from an earlier version of a WebSphere MQ for AIX client to IBM WebSphere MQ for AIX Version 7.0 clients. If you are migrating from an earlier version of WebSphere MQ or MQSeries® for AIX, you are advised to uninstall your current version before installing the IBM WebSphere MQ for AIX Version 7.0 client.

Migration from an earlier version of WebSphere MQ for AIX involves updating any currently installed filesets, and installing any new filesets that might be required.

To update currently installed filesets:

1. Go into SMIT for root authority. From the shell, enter:
   ```
   smit
   ```
2. Select the device appropriate for your installation using the following sequence of windows:
   ```
       Software Installation and Maintenance
         Install and Update Software
           Update Installed Software to Latest Level (Update All)
   ```
   Alternatively, you can use the fastpath command to select the appropriate device:
   ```
   smitty update_latest
   ```
3. Select the **List** button to display the Single Select List window.
4. Select **/dev/cd0 (CD Drive)**.
5. Select **OK** to display the parameters for **Update All**.
6. Update all previously installed software for WebSphere MQ by selecting the **_update_all** option in the **Software to update** field.
7. Press Enter.
8. Select **OK** in the confirmation window to start updating the software.

When all previously installed filesets have been updated to the latest level, you can install any additional filesets. See the installation instructions in *WebSphere MQ for AIX Quick Beginnings* for more information.

Changes to the installation path:

The current version of the WebSphere MQ for AIX client installs into a different directory to versions earlier than Version 5.3. If you are migrating from an early version, you need to take action.

Changes in AIX LPP Version 4 packaging mean that the IBM WebSphere MQ for AIX, Version 5.3 and later clients install into directory **/usr/mqm**. Versions previous to WebSphere MQ for AIX, V5.3 of the product installed into directory **/usr/lpp/mqm**.

Installation of the IBM WebSphere MQ for AIX client fails if a file system mounted as **/usr/lpp/mqm** is detected. If you are migrating from an earlier version and a file system exists for this directory, you will need to do one of the following things before installing the IBM WebSphere MQ for AIX client. Either:

- Uninstall your existing WebSphere MQ or MQSeries client, and either delete the file system or move it to the new install path of **/usr/mqm**

or

- Move the old file system of **/usr/lpp/mqm** to the new install path of **/usr/mqm** and create a symbolic link from the old path to the new by issuing the following command:

```
ln -s /usr/mqm /usr/lpp/mqm
```

If you uninstall your existing client and either delete or move your existing file system, you can then install the IBM WebSphere MQ for AIX client as described in the installation instructions.

However, if you move the old file system to the new installation path, you should then perform the migration installation described in “Migrating from an earlier version of WebSphere MQ for AIX client” on page 27.

Note: If you have already symbolically linked a file system to **/usr/lpp/mqm**, installation of the IBM WebSphere MQ for AIX client will destroy the contents of the file system and the symbolic link, leaving an empty file system. If this happens, you are advised to uninstall your existing WebSphere MQ client and either delete the file system or relink it to the new install path of **/usr/mqm**, before installing the IBM WebSphere MQ for AIX client.

The installation process for the IBM WebSphere MQ for AIX client creates a symbolic link from the old install path (**/usr/lpp/mqm**) to the new install path (**/usr/mqm**). Therefore any existing scripts or makefiles that reference the old path are still valid.

Changing the national language

Installation defaults to the national language that was specified when your operating system was installed.

First, check the initial locale setting for your machine by typing:

```
smitty mle_cc_cust_hdr
```

and press Enter. If this is not one of the national languages provided by WebSphere MQ, you must select a national language, otherwise you will not get a message catalog installed on your system.

It is possible to install the WebSphere MQ client software so that the online help and messages are in another national language. For instructions on how to do this, see WebSphere MQ for AIX Quick Beginnings.

Removing a WebSphere MQ client from AIX

Use SMIT as follows to remove all the WebSphere MQ client files that were installed.

1. Type `smit`
2. Follow this sequence of windows:
   ```
   Software Installation and Maintenance
     Software Maintenance and Utilities
       Remove Installed Software
   ```
3. Press the **List** button to display the installed software and select the filesets to remove. The WebSphere MQ client filesets have titles starting mqm*.

Migrating to and from the WebSphere MQ SSL support

To upgrade a WebSphere MQ client without the SSL support to one with the SSL support, install the three additional file sets, gsksa.rte, gskta.rte, and mqm.keyman.rte, from WebSphere MQ Client CD 1. Follow the installation instructions, choosing only those file sets. To downgrade a WebSphere MQ client with the SSL support to one without the SSL support, remove these three file sets.

Uninstalling the extended transactional function

This section describes how to uninstall, or remove, the extended transactional function, leaving the WebSphere MQ base client on your system.

For information about how to uninstall the WebSphere MQ base client after you have uninstalled the extended transactional function, see “Removing a WebSphere MQ client from AIX.”

Before you attempt to uninstall the extended transactional function, end all WebSphere MQ activity on your system by stopping any of the following that might be running:

- WebSphere MQ applications
- Transaction managers that are using queue managers as resource managers
- Queue managers and their associated listeners

To uninstall the extended transactional function, follow this procedure, which uses the **installp** command:

1. Log in as root.
2. Enter the following command:
   ```
   installp -u mqm.txclient.rte
   ```
   The uninstallation now runs to completion.

Installing on HP-UX

To install the WebSphere MQ client on an HP-UX system, use WebSphere MQ Client CD 2, which is supplied with WebSphere MQ, and follow the instructions in WebSphere MQ for HP-UX Quick Beginnings.

Note: If you plan to install a WebSphere MQ client and server on the same machine, see "Installing WebSphere MQ clients on the same machine as the server" on page 24.

The WebSphere MQ client is installed into the `/opt/mqm` directory. This *cannot* be changed.

To install the extended transactional client on HP-UX, you must first install the WebSphere MQ base client. As a minimum, you must install the runtime, client, and client for Java components of the WebSphere MQ base client.

Components for HP-UX

The components you can install on HP-UX systems are:

WebSphere MQ Client
The WebSphere MQ client code for your UNIX platform.

Samples
Sample application programs.

Runtime component
Support for external applications. This does **not** enable you to write your own applications.

Base
Support to enable you to create and support your own applications. Requires the runtime component to be installed.

WebSphere MQ Client for Java
This allows Java applets running on your client machine to communicate with WebSphere MQ. It includes security exits for encryption and authentication of messages sent across the Web by the WebSphere MQ Client for Java. These exits consist of some Java classes. To use the client for Java you need to have Java runtime code on your machine, at the following (or later compatible) levels:

HP-UX (PA-RISC platform)

- 32 bit
  - HP-UX SDK for the Java platform and JDK V1.4
  - HP SDK for J2SE HP-UX 11i platform, adapted by IBM for IBM software V1.4.2
  - HP SDK for J2SE HP-UX 11i platform, adapted by IBM for IBM software V5.0 (SR1 or above)
  - HP SDK for J2SE HP-UX 11i platform, adapted by IBM for IBM software V6.0
- 64 bit:
  - HP-UX SDK for the Java platform and JDK V1.4
  - IBM 64-bit SDK for HP-UX Java 2 Technology Edition V1.4.2
  - IBM 64-bit SDK for HP-UX Java 2 Technology Edition V5.0 (SR1 or above)
  - IBM 64-bit SDK for HP-UX Java 2 Technology Edition V6.0 (SR1 or above)

Figure 16. Java

HP-UX (Itanium platform)

- 32-bit
  - HP-UX IPF Software Development Kit for the Java™ 2 platform
  - HP SDK for J2SE HP-UX 11i platform, adapted by IBM® for IBM Software, V1.4.2 for 32-bit Itanium
  - HP SDK for J2SE HP-UX 11i platform, adapted by IBM for IBM Software, V5.0 for 32-bit Itanium (SR1 or above)
  - HP SDK for J2SE HP-UX 11i platform, adapted by IBM for IBM Software, V6.0 for 32-bit Itanium
- 64-bit
  - HP-UX IPF Software Development Kit for the Java 2 platform
  - HP SDK for J2SE HP-UX 11i platform, adapted by IBM for IBM Software, V1.4.2 for 64-bit Itanium
  - HP SDK for J2SE HP-UX 11i platform, adapted by IBM for IBM Software, V5.0 for 64-bit Itanium (SR1 or above)
  - HP SDK for J2SE HP-UX 11i platform, adapted by IBM for IBM Software, V6.0 for 64-bit Itanium

Figure 17. Java

For information about Java runtime, see the WebSphere MQ Using Java book.

Note: If it is possible on your platform, at installation time the CLASSPATH environment variable will either be updated if already present, or created if not.

Installing the extended transactional function

This section describes how to install the extended transactional function using the **swinstall** command.

Before you attempt to install the extended transactional function, end all WebSphere MQ activity on your system by stopping any of the following that might be running:

- WebSphere MQ applications
- Queue managers and their associated listeners

To install the extended transactional function, follow this procedure:

1. Log in as root.
2. Insert the WebSphere MQ Extended Transactional Clients CD-ROM into the CD-ROM drive and mount it on /cdrom, or a directory of your choice. If you use a different directory, remember to use the name of that directory, instead of /cdrom, in the instructions that follow.

   To mount the CD-ROM on /cdrom, follow this procedure:

   a. Enter `cd /usr/sbin`
   b. Enter `pfs_mountd &`
   c. Enter `pfsd 4 &`
   d. Enter `pfs_mount -o xlat=unix /<path to CD-ROM device> /cdrom`
3. Run the mqlicense.sh script by entering the following command:
   ```
   /cdrom/hpux11/mqlicense.sh
   ```
   To view the license in a format that can be read by a screen reader, enter the following command instead:
   ```
   /cdrom/hpux11/mqlicense.sh -text_only
   ```
   The license is displayed. If you accept the license, you can continue the installation. If you do not accept the license, you cannot continue.
4. To start the installation process, enter the following command:
   ```
   swinstall -s src_dir MQSERIES.MQM-TXCLIENT
   ```

where *src_dir* is the source directory for the installation files (similar to: /cdrom/hpux11/mq...).

Removing a WebSphere MQ client from HP-UX

To remove a WebSphere MQ client from your HP-UX system, use the **swremove** command, or use SAM. You can then delete the /var/mqm directory tree.

Migrating to and from the WebSphere MQ SSL support

To upgrade a WebSphere MQ client without the SSL support to one with the SSL support, install the two additional file sets, gsk7bas and MQSERIES.MQM-KEYMAN, from the directory on WebSphere MQ Client CD 2 that contains the set of client components with the WebSphere MQ SSL support.

Follow the installation instructions, selecting only those file sets. To downgrade a WebSphere MQ client with the SSL support to one without the SSL support, remove these two file sets.

Uninstalling the extended transactional function

This section describes how to uninstall, or remove, the extended transactional function, leaving the WebSphere MQ base client on your system.

For information about how to uninstall the WebSphere MQ base client after you have uninstalled the extended transactional function, see “Removing a WebSphere MQ client from HP-UX.”

Before you attempt to uninstall the extended transactional function, end all WebSphere MQ activity on your system by stopping any of the following that might be running:

- WebSphere MQ applications
- Transaction managers that are using queue managers as resource managers
- Queue managers and their associated listeners

To uninstall the extended transactional function, follow this procedure, which uses the **swremove** command:

1. Log in as root.
2. Enter the following command:
   ```
   swremove MQSERIES.MQM-TXCLIENT
   ```
   The uninstallation now runs to completion.

Installing on Linux

To install the WebSphere MQ client on a Linux system, use WebSphere MQ Client CD 3, which is supplied with WebSphere MQ, and follow the instructions in WebSphere MQ for Linux Quick Beginnings.

If you plan to install the WebSphere MQ client and on the same machine as the WebSphere MQ server, see “Installing WebSphere MQ clients on the same machine as the server” on page 24.

To install the extended transactional client on Linux, you must first install the WebSphere MQ base client. As a minimum, you must install the runtime, client, and client for Java components of the WebSphere MQ base client.

Installing the extended transactional function

Before you attempt to install the extended transactional function, end all WebSphere MQ activity on your system by stopping any of the following that might be running:

- WebSphere MQ applications
- Queue managers and their associated listeners

To install the extended transactional function, follow this procedure, which uses the Red Hat Package Manager (RPM) installer.

1. Log in as root.
2. Insert the WebSphere MQ Extended Transactional Clients CD-ROM into the CD-ROM drive and mount it on /cdrom, or a directory of your choice. If you use a different directory, remember to use the name of that directory, instead of /cdrom, in the instructions that follow.
3. If you are installing on an Intel machine, change to the /cdrom/linux_intel directory.

   If you are installing on a zSeries machine, change to the /cdrom/linux_zseries directory

   If you are installing on a POWER machine, change to the /cdrom/linux_power directory

   Note:

   a. If your machine does not have a locally attached CD-ROM drive, you can copy the contents of this directory from a machine that does have a CD-ROM drive to your machine using, for example, the ftp utility. You can then install the extended transactional function from your local copy of the directory.
   b. If the machine hosting the CD-ROM is an NFS server, you can mount the contents of the CD-ROM on a directory on your system using NFS.
4. Run the mqlicense.sh script by entering the following command:
   ```
   ./mqlicense.sh
   ```
   To view the license in a format that can be read by a screen reader, enter the following command instead:
   ```
   ./mqlicense.sh -text_only
   ```
   The license is displayed. If you accept the license, you can continue the installation. If you do not accept the license, you cannot continue.
5. Use the **rpm -ivh** command to install the package containing the extended transactional function. For example: on x86 architecture, enter the following command:
   ```
   rpm -ivh MQSeriesTXClient-7.0.0-0.i386.rpm
   ```

Removing the WebSphere MQ client from Linux

Before you attempt to remove the WebSphere MQ client, check that no WebSphere MQ client application is running on your system.

To remove the WebSphere MQ client, you must first find out the package names of the components currently installed on your system. To list the package names with their version information, enter the following commands:

```
rpm -q -a | grep MQ
rpm -q -a | grep gsk
```

Alternatively, to list the package names without their version information, enter the following commands:

```
rpm -q -a --queryformat "%{NAME}\n" | grep MQ
rpm -q -a --queryformat "%{NAME}\n" | grep gsk
```

To remove a component, with package name `MQSeriesSamples` for example, enter the following command:

```
rpm -e MQSeriesSamples
```

Some of the components are dependent on others. The **rpm** command does not remove a component if others are dependent on it. For this reason, you must remove the components in an order such that each component you remove has no other component dependent on it. To list all the components on which a specific component depends, `MQSeriesClient` for example, enter the following command:

```
rpm -q --requires MQSeriesClient
```

Alternatively, remove the components in the order shown in Table 4. Remove only those components that you have installed on your system.

Table 4. Order for removing components

Component	**Package name**
Message catalogs	`MQSeriesMsg_`*xx*[1]
Sample programs	`MQSeriesSamples`
Client	`MQSeriesClient`
SDK	`MQSeriesSDK`
Runtime	`MQSeriesRuntime`
WebSphere MQ support for iKeyman	`MQSeriesKeyMan`
IBM Global Security Kit (GSKit)	`gsk7bas`[2]

Note:
1. *xx* identifies the national language.
2. Other IBM products might use the IBM Global Security Kit.

After removing the WebSphere MQ client, delete the `/var/mqm` directory, unless you are migrating to a later release of the WebSphere MQ client.

Migrating to and from the WebSphere MQ SSL support

To upgrade a WebSphere MQ client without the SSL support to one with the SSL support, install the two additional components, IBM Global Security Kit (GSKit) and WebSphere MQ support for iKeyman.

Follow the instructions for installation, selecting only those two components. To downgrade a WebSphere MQ client with the SSL support to one without the SSL support, remove these two components.

Uninstalling the extended transactional function

This section describes how to uninstall, or remove, the extended transactional function, leaving the WebSphere MQ base client on your system.

For information about how to uninstall the WebSphere MQ base client after you have uninstalled the extended transactional function, see “Removing the WebSphere MQ client from Linux” on page 33.

Before you attempt to uninstall the extended transactional function, end all WebSphere MQ activity on your system by stopping any of the following that might be running:

- WebSphere MQ applications
- Transaction managers that are using queue managers as resource managers
- Queue managers and their associated listeners

To uninstall the extended transactional function, enter the following command:

```
rpm -e MQSeriesTXClient
```

Installing on Solaris

To install the WebSphere MQ client on a Solaris system, use WebSphere MQ Client CD 4, which is supplied with WebSphere MQ, and follow the instructions in WebSphere MQ for Solaris Quick Beginnings. Note that you should log in as root and the root userid's umask setting must be 022.

Note: If you plan to install a WebSphere MQ client and server on the same machine, see "Installing WebSphere MQ clients on the same machine as the server" on page 24.

To install the extended transactional client on Solaris, you must first install the WebSphere MQ base client. As a minimum, you must install the client libraries component of the WebSphere MQ base client.

Components for Solaris

The components you can install on Solaris systems are:

WebSphere MQ Client
The WebSphere MQ client code for your UNIX platform.

Samples
Sample application programs.

Runtime component
Support for external applications. This does **not** enable you to write your own applications.

Base
Support to enable you to create and support your own applications. Requires the runtime component to be installed.

WebSphere MQ Client for Java
This allows Java applets running on your client machine to communicate with WebSphere MQ. It includes security exits for encryption and authentication of messages sent across the Web by the WebSphere MQ Client for Java. These exits consist of some Java classes. To use the client for Java you need to have Java runtime code on your machine, at the following (or later compatible) levels:

Solaris (SPARC platform)

- 32-bit
  - Sun Solaris Java SDK V1.4 with JDK V1.4.2
  - Sun Solaris Java SDK V5 with JDK V5
  - Sun Solaris Java SDK V6 with JDK V6
  - IBM 32-bit SDK for Solaris, Java 2 Technology Edition V1.4.2
  - IBM 32-bit SDK for Solaris, Java 2 Technology Edition V5.0 (SR1 or above)
  - IBM 32-bit SDK for Solaris, Java 2 Technology Edition V6.0
- 64-bit
  - Sun Java 2 SDK, Standard Edition V1.4.2
  - Sun Java 2 SDK, Standard Edition V5
  - Sun Java 2 SDK, Standard Edition V6
  - IBM® 64-bit SDK for Solaris, Java 2 Technology Edition, Version 1.4.2
  - IBM 64-bit SDK for Solaris, Java 2 Technology Edition V5.0 (SR1 or above)
  - IBM 64-bit SDK for Solaris, Java 2 Technology Edition V6.0

Figure 18. Java

Solaris (x86-64 platform)

- 32-bit
  - Sun Java 2 Platform Standard Edition, Version 1.4.2
  - Sun Java 2 Platform Standard Edition, Version 5.0
  - Sun Java 2 Platform Standard Edition, Version 6.0
  - IBM 32-bit SDK for Solaris, Java 2 Technology Edition, Version 1.4.2
  - IBM 32-bit SDK for Solaris, Java 2 Technology Edition V5.0 (SR1 or above)
  - IBM 32-bit SDK for Solaris, Java 2 Technology Edition V6.0
- 64-bit
  - Sun Java 2 Platform Standard Edition, Version 5.0
  - Sun Java 2 Platform Standard Edition, Version 6.0
  - IBM 64-bit SDK for Solaris, Java 2 Technology Edition V5.0 (SR1 or above)
  - IBM 64-bit SDK for Solaris, Java 2 Technology Edition V6.0

Figure 19. Java

For information about Java runtime see WebSphere MQ Using Java.

Note: If it is possible on your platform, at installation time the CLASSPATH environment variable will either be updated if already present, or created if not.

Installing the extended transactional function

This section describes how to install the extended transactional function using the **pkgadd** command.

Note: If you are using a screen reader, you might want to install the extended transactional function in unattended, or silent mode, so that you can accept the license without viewing it. See “Unattended, or silent, installation” on page 37 for information about how to do this.

Before you attempt to install the extended transactional function, end all WebSphere MQ activity on your system by stopping any of the following that might be running:

- WebSphere MQ applications
- Queue managers and their associated listeners

To install the extended transactional function, follow this procedure:

1. Log in as root.
2. Insert the WebSphere MQ Extended Transactional Clients CD-ROM into the CD-ROM drive and mount it on /cdrom, or a directory of your choice. If you use a different directory, remember to use the name of that directory, instead of /cdrom, in the instructions that follow.
3. Run the mqlicense.sh script by entering the following command:
   ```
   /cdrom/solaris/mqlicense.sh
   ```
   To view the license in a format that can be read by a screen reader, enter the following command instead:
   ```
   /cdrom/solaris/mqlicense.sh -text_only
   ```
   The license is displayed. If you accept the license, you can continue the installation. If you do not accept the license, you cannot continue.
4. To start the installation process, enter a command similar to the following:
   ```
    pkgadd -d /cdrom/solaris/src_name
   ```
   where *src_name* is the filename of the package to install (similar to mq...img). A list containing only one installable package, mqm-txcli, is displayed.
5. Enter `1` or `all`.
6. Answer `y` to all subsequent questions.

   When the installation completes successfully, the following message is displayed:
   ```
   Installation of <mqm-txcli> was successful.
   ```

Unattended, or silent, installation:

You can install the extended transactional function on a system without any interaction. This process is called an unattended, or silent, installation.

A script file, `silent.sh`, is supplied in the /cdrom/solaris/silent directory. The script uses two other files, `admin` and `response`, which are supplied in the same directory.

The script assumes that the WebSphere MQ Extended Transactional Clients CD-ROM is mounted on /cdrom, and it directs all screen output and log information to the file `/tmp/mq.install`. If you want to change the script, copy the `silent.sh`, `admin`, and `response` files to a directory on your system, make the required changes, and run the script from that directory. When the installation is complete, check the log information for any errors.

Removing a WebSphere MQ client from Solaris

If you have previously installed WebSphere MQ on your system, you must remove the product using the **pkgrm** program.

If the product is present, but not installed correctly, you might need manually to delete the files and directories contained in:

```
/var/mqm
/opt/mqm
```

Migrating to and from the WebSphere MQ SSL support

To upgrade a WebSphere MQ client without the SSL support to one with the SSL support, install the image from the directory on WebSphere MQ Client CD 4 that contains the set of client components with the WebSphere MQ SSL support (see the installation instructions).

When you are asked whether you really want to install the image, answer "yes".

To downgrade a WebSphere MQ client with the SSL support to one without the SSL support, remove all the components of the client and install the client again, this time using the set of client components without the WebSphere MQ SSL support.

Uninstalling the extended transactional function

This section describes how to uninstall, or remove, the extended transactional function, leaving the WebSphere MQ base client on your system.

For information about how to uninstall the WebSphere MQ base client after you have uninstalled the extended transactional function, see "Removing a WebSphere MQ client from Solaris" on page 37.

Before you attempt to uninstall the extended transactional function, end all WebSphere MQ activity on your system by stopping any of the following that might be running:

- WebSphere MQ applications
- Transaction managers that are using queue managers as resource managers
- Queue managers and their associated listeners

To uninstall the extended transactional function, follow this procedure, which uses the **pkgrm** command:

1. Log in as root.
2. Enter the following command:
   ```
   pkgrm mqm-txcli
   ```
3. Answer `y` to all subsequent questions.

   When the uninstallation completes successfully, the following message is displayed:
   ```
   Removal of <mqm-txcli> was successful.
   ```

Installing on Windows

See the following sections for information on how to install the WebSphere MQ client on Windows:

- "Preparing to install the WebSphere MQ client" on page 39
- "Installing the WebSphere MQ client" on page 39
- "Installing from a LAN" on page 43
- "Unattended (silent) installation" on page 45
- "Advanced installation methods" on page 47
- "Using Microsoft System Management Server" on page 55
- "Uninstalling a WebSphere MQ client" on page 56

To install the WebSphere MQ client on Windows, use WebSphere MQ Client CD 5, which is supplied with WebSphere MQ.

To install the extended transactional client on Windows systems, you must first install the WebSphere MQ base client.

Preparing to install the WebSphere MQ client

Before you install, decide what type of installation you require.

Table 5 shows the installation types available, and the features that are installed with each option. For the prerequisites required for each feature, see "Windows client: hardware and software required" on page 20.

Table 5. Features installed with each type of installation

Installation type	Features installed	Comments
Typical	• Windows Client • Java Messaging and SOAP transport • Development Toolkit	The default option. Features are installed to default locations.
Compact	• Windows Client only	The feature is installed to the default location.
Custom	By default, the following features are preselected: • Windows Client • Java Messaging and SOAP transport • Development Toolkit	All the available features are listed and you can select which ones to install, and where to install them.

Installing the WebSphere MQ client

To install a WebSphere MQ client, you must be logged on to Windows as an administrator.

WebSphere MQ checks for any existing WebSphere MQ configuration files (MQS.INI). If it finds any, it automatically migrates configuration information to the Windows Registry. Otherwise, WebSphere MQ automatically puts its configuration information directly into the Windows Registry.

Note: When installing using a Remote Desktop Connection, you might need to logoff, then re-logon to pick up the changes made to your environment by the installation process.

Typical client installation:

The following instructions assume that you are installing the WebSphere MQ client using WebSphere MQ Client CD 5, which is supplied with WebSphere MQ. If you plan to install a WebSphere MQ client and server on the same machine, see the WebSphere MQ for Windows Quick Beginnings book.

1. Insert WebSphere MQ Client CD 5 into the CD drive.

   If autorun is enabled, the installation process starts. If it is not, double-click the **Setup** icon in the root folder on the CD to start the process.

   The Select Setup Language window is displayed.
2. On the Select Setup Language window, select the national language of your choice from the list, then click **OK**.

   The WebSphere MQ Client Setup window is displayed.

3. Click **Next** to continue.

   If the current version of WebSphere MQ client is already installed, the Program Maintenance panel is displayed with two options: Modify or Remove.

   a. If you select Modify, see “Modifying the client installation” on page 41.
   b. If you select Remove, see “Uninstalling WebSphere MQ client using the installation process” on page 57.

   If the current version of WebSphere MQ client is not installed, the License Agreement panel is displayed.
4. Read the information and license terms on the panel.

   To change the language that the license agreement is displayed in, click **Change Language** then select the language you require from the list provided.

   Select the option to accept the license terms, then click **Next**.
5. If there was no previous version of this product installed on the machine, the Setup Type panel is displayed.

   Select the type of installation you want, then click **Next**. Table 5 on page 39 shows the installation types and the features that are installed with each option.

   a. If you select Custom, go to the procedure “Custom client installation” on page 41.
   b. If you select Typical or Compact, go to step 7.
6. If there was a previous version of WebSphere MQ installed on the machine, the Type of Installation Process panel is displayed. Select one of the following options, then click **Next**:
   - Update. Installs the same features as the previous version. Go to the next step.
   - Custom. You can select which features to install.

     If you select this option, a Destination Folders panel for data files is displayed, then the Features panel is displayed. Follow the procedure “Custom client installation” on page 41 from step 3 on page 41 or 4 on page 41 as appropriate.
7. The WebSphere MQ Client Setup window displays a summary of the installation you selected.

   To continue, click **Install**.
8. Wait until the progress bar is complete.

   When the WebSphere MQ client is successfully installed, the WebSphere MQ Client Setup window displays the following message:

   Installation Wizard Completed Successfully

   Click **Finish** to close the window.
9. At this point run the Prepare WebSphere MQ Wizard. This will assist you in migrating any SSL certificates that you may have.
10. The installation of the WebSphere MQ client is now complete. Note that WebSphere MQ clients are sets of services and do not have to be explicitly run.
11. You now need to verify that the client was installed successfully (see “Verifying the installation” on page 75).

Compact client installation:

Follow the steps for a typical client installation, as described in "Typical client installation" on page 39. The only difference is that, at step 5 on page 40, you select **Compact** on the **Setup Type** window. This installs only the Client feature of WebSphere MQ for Windows.

Custom client installation:

During custom installation, you can choose the destination folders for program files and data files. However, after installation, you cannot change these (except by removing the product, then reinstalling). Therefore, plan and select your destination folders carefully.

1. Follow steps 1 on page 39 to 5 on page 40 of the "Typical client installation" on page 39.
2. At step 5 on page 40, select **Custom** on the **Setup Type** window.
3. The Destination Folder panel is displayed.

   To accept the default folder for the program files, select **Next**.

   To change the folder for the program files, select **Change**, select the required folder in the resulting dialog box, select **OK**, then select **Next**.
4. The Destination Folders panel is displayed.

   To accept the default folder for the data files, select **Next**.

   To change the folder for the data files, select **Change**, select the required folder in the resulting dialog box, select **OK**, then select **Next**.

   If you want to install the Client , you require a data files folder. Otherwise, you can ignore this panel (that is, accept the default).
5. The Features panel is displayed.
6. To change the installation of a feature:
   a. Click the symbol to the left of the feature name to display a drop-down menu.
   b. Select the required option from:
      - Install this feature
      - Install this feature and all its subfeatures (if any)
      - Do not install this feature (remove if already installed)

      The symbol to the left of the feature name changes to show the current installation option. For more information, click **Help** to display the Custom Setup Tips page, which explains the icons used in the feature list.
7. Optionally, to check that there is enough disk space, press the **Space bar**.

   The Disk Space Requirements panel is displayed. This shows the disk space available and the amount of disk space that your current selections will take. It highlights any volumes that do not have enough disk space.

   To close the panel and return to the Features panel, click **OK**.
8. When your selections are complete, click **Next**.
9. Follow from step 7 on page 40 to the final step of the procedure.

Modifying the client installation:

You modify the installation when WebSphere MQ for Windows client is installed and you want to remove or install some WebSphere MQ client features.

1. Insert WebSphere MQ Client CD 5 into the CD drive.
2. If autorun is installed, the installation process starts.

Otherwise, double-click on the Setup icon in the root folder of the CD to start the installation process.

The WebSphere MQ Client Setup window is displayed. Click **Next** to continue.

3. Select **Modify**, then click **Next**.

   The Features panel is displayed.

4. To change the installation of a feature:
   a. Click on the symbol to the left of the feature name to display a drop-down menu.
   b. Select the required option from:
      - Install this feature
      - Install this feature and all its subfeatures (if any)
      - Do not install this feature (remove if already installed).

      The symbol to the left of the feature name changes to show the current installation option.

5. When your selections are complete, click **Next**.

6. The WebSphere MQ Client Setup window displays a summary of the installation you selected.

   To continue, click **Modify**.

7. Wait until the progress bar is complete.

   When the WebSphere MQ client is successfully installed, the WebSphere MQ Client Setup window displays the following message:

   Installation Wizard Completed Successfully

   Click **Finish** to close the window.

Modifying the client installation using Add/Remove Programs:

For Windows 2000, Windows XP, or Windows 2003, follow these steps. You cannot use this method to modify an installation on Windows Vista or Windows Server 2008:

1. From the Windows taskbar, select **Start** → **Settings** → **Control Panel**.
2. Select **Add/Remove Programs**.
3. Select **IBM WebSphere MQ**.
4. Select **Change**.

   The WebSphere MQ Setup window with the Program Maintenance panel is displayed. Follow the procedure for modifying the installation using the process from step 3 to the end.

Installing the extended transactional function:

To install the extended transactional function, follow this procedure. A "Typical client installation" on page 39 is required.

1. Insert the WebSphere MQ Server CD-ROM into the CD-ROM drive.

   If autorun is enabled, the installation process starts. If it is not enabled, double-click the **Setup** icon in the Windows folder on the CD-ROM to start the installation process.

   The Select Setup Language window opens.

2. If autorun is installed, the installation process starts.

   Otherwise, double-click on the Setup icon in the root folder of the CD to start the installation process.

The WebSphere MQ Client Setup window is displayed. Click **Next** to continue.

3. Select **Modify**, then click **Next**.

   The Features panel is displayed.
4. Make sure the extended transactional function is selected to be installed:
   a. Click on the symbol to the left of **Client Extended Transaction Support** to display a drop-down menu.
   b. Select **Install this feature**. The symbol to the left changes to show the new installation option.

   Note: Ensure the following features are also selected to be installed:
   - Windows Client
   - Java Messaging and Web Services
   - Development Toolkit
5. When your selections are complete, click **Next**.
6. The WebSphere MQ Client Setup window displays a summary of the installation you selected.

   To continue, click **Modify**.
7. Wait until the progress bar is complete.

   When the WebSphere MQ client is successfully installed, the WebSphere MQ Client Setup window displays the following message:

   Installation Wizard Completed Successfully

   Click **Finish** to close the window.

Other methods of installing the WebSphere MQ client

This section contains instructions on how to install the WebSphere MQ client from a LAN, and how to install the WebSphere MQ client using System Management Server (SMS).

Installing from a LAN:

There are two ways to put WebSphere MQ installation files on a LAN server for easier access:

- You can make the drive, into which the WebSphere MQ Client CD is inserted, shareable.
- You can copy the installation files from the CD to a server. To do this, use the following steps:
  1. Create a folder on the LAN server to store the installation files. For example:
     ```
     md m:\instmqc
     ```
  2. Load WebSphere MQ Client CD 5. If autorun is enabled, the WebSphere MQ language_selection window is displayed. Select **Cancel** to close this window.
  3. Copy the entire CD to the installation folder. For example:
     ```
     xcopy e:\*.* m:\instmqc /e
     ```
  4. Give all licensed users access to the folder that now contains the CD image (in this example, the `m:` drive).
  5. From a command prompt on the target machine, type the following:
     ```
     \\servername\installation_folder\Windows\setup.exe
     ```

     where *servername* is the name of the server and *installation_folder* is the full path of the installation folder.

     Alternatively:

a. Map the shared resource to a drive letter. You can use the net use command, or the Windows Explorer.
b. Change to the installation folder.
c. Type `setup`, then press Enter.

6. Follow the prompts.

Installing the extended transactional client from a LAN server:

To install the extended transactional function from a LAN server, you must first make the installation files accessible on a target system. There are two ways of doing this:

- On the LAN server, create a share name for the drive into which the WebSphere MQ Extended Transactional Clients CD-ROM is inserted. Give all licensed users access to drive.
- Copy the installation files from the CD-ROM to a folder on the LAN server and make the folder shareable. To do this, use the following procedure:
  1. Create a folder on the LAN server to store the installation files. For example, enter the following command at a command prompt:
     ```
     md m:\instmqc
     ```
  2. Insert the WebSphere MQ Extended Transactional Clients CD-ROM into the CD-ROM drive.

     If autorun is enabled, the Select Setup Language window opens. Click **Cancel** to close this window.
  3. Copy the contents of the CD-ROM to the installation folder. For example, enter the following command at a command prompt:
     ```
     xcopy e:\*.* m:\instmqc\ /e
     ```
     Alternatively, if you want to copy only the directories required for installing on Windows systems, enter the following commands at a command prompt:
     ```
     xcopy e:\Windows\*.* m:\instmqc\Windows\ /e
     xcopy e:\Readmes\Windows\*.* m:\instmqc\Readmes\Windows\ /e
     xcopy e:\Licenses\Windows\*.* m:\instmqc\Licenses\Windows\ /e
     ```
  4. Create a share name for the installation folder and give all licensed users access to the folder.

You can now use the following procedure to install the extended transactional function:

1. From a command prompt on a target system, enter the following command:
   ```
   \\server_name\share_name\Windows\setup.exe
   ```
   where *server_name* is the name of the LAN server and *share_name* is the share name of the CD-ROM drive or installation folder on the LAN server.

   Alternatively:
   a. Map \\*server_name**share_name* to a drive letter using the **net use** command or Windows Explorer.
   b. At a command prompt, change to the drive letter, and then to the Windows directory within the drive.
   c. Type `setup`, and press Enter.

   The Select Setup Language window opens.
2. Follow the procedure in "Installing the extended transactional function" on page 42 from step 3 on page 43 to the end.

Unattended (silent) installation:

WebSphere MQ for Windows client is installed using the Microsoft Installer (MSI). You can invoke MSI directly, without using setup.exe.

This means that you can install WebSphere MQ on a machine without interaction. This process is called unattended (or silent) installation, and is particularly useful for installing WebSphere MQ over a network on a remote machine, because you can install from a shared drive on a LAN server.

If you are running WebSphere MQ on Windows Vista or Windows Server 2008 with User Account Control (UAC) enabled, you must invoke the silent installation from an elevated command prompt. Elevate a command prompt by using a right-click to start the command prompt and choose **Run as administrator**. If you try to silently install from a non-elevated command prompt, the install fails with an error of AMQ4353 in the install log.

To do this, you can invoke MSI with a parameter that calls a response file. A response file is an ASCII text file that contains the parameter values you want to set for the installation.

The machine on which you want to install must be able to share the WebSphere MQ Client CD, or a copy of the files on it, and you must be able to execute a command on that machine.

Note:

1. The response file you use to install WebSphere MQ for Windows using the WebSphere MQ Client CD is *not* the same as the one used with earlier non-MSI versions of MQSeries. For details about the response file you use with WebSphere MQ Client CD, see "Using msiexec with a response file" on page 52.
2. There are several other methods to invoke MSI without setup.exe. For details, see "Advanced installation methods" on page 47.

To invoke a silent installation using a response file, you use the `msiexec` command.

The response file is an ASCII text file, with a format similar to a Windows .ini file, that contains the stanza **[Response]**. This stanza contains parameters that the `msiexec` command can use, in the form of PROPERTY=value pairs. The `msiexec` command ignores any other stanzas in the file. An example response file, Response.ini, is supplied with WebSphere MQ. This file contains default installation parameters.

There are three ways to create a response file for installation:

- Copy and edit the file Response.ini that is supplied on WebSphere MQ Client CD 5, using an ASCII file editor.
- Create your own response file using an ASCII file editor.
- Use an advanced method to invoke an installation and specify the SAVEINI property (and optionally, the ONLYINI property) to generate a response file that contains the same installation options. For more information, see Creating a response file.

In the response file, all text is in English, and comments begin with a ; character.

Invoking a silent installation:

To invoke a typical silent installation, enter the following command at a command line:

```
msiexec /i "path\MSI\IBM WebSphere MQ.msi" /q
TRANSFORMS="1033.mst" AGREETOLICENSE="yes"
```

where:

/q Specifies a silent installation.

If you are running WebSphere MQ on Windows Vista or Windows Server 2008 with User Account Control (UAC) enabled, you must invoke the silent installation from an elevated command prompt. Elevate a command prompt by using a right-click to start the command prompt and choose **Run as administrator**. If you try to silently install from a non-elevated command prompt, the install fails with an error of AMQ4353 in the install log.

TRANSFORMS=″1033.mst″ specifies that the installation is in US English. For further information about installing in different national languages, see "Using transforms with msiexec" on page 51.

You can also specify property=value pairs on the command line (the property must be in upper case), for example:

```
msiexec /i "path\MSI\IBM WebSphere MQ.msi" /q ADDLOCAL="Client"
TRANSFORMS="1033.mst" AGREETOLICENSE="yes"
```

- PROPERTY strings must be in upper case.
- Value strings are not case-sensitive, except for feature names. They can be enclosed in double quotation marks. If a value string includes a blank, enclose the blank in double quotation marks.
- For a property that can take more than one value, use the format:

  ```
  ADDLOCAL="Client,Toolkit"
  ```

Table 6. Valid feature names for the ADDLOCAL and REMOVE properties

Feature Name	Description
Client	The WebSphere MQ for Windows client.
JavaMsg	Java Messaging and SOAP transport support
Toolkit	Sample WebSphere MQ program source and sample executable code.

An example of a typical response file is:

```
[Response]
PGMFOLDER="c:\mqm"
DATFOLDER="c:\mqm\data"
AGREETOLICENSE="yes"
ADDLOCAL="Client"
REMOVE="Toolkit"
```

For a full list of response file parameters see Using a response file with msiexec.

Unattended installation of the extended transactional client:

The extended transactional function is installed using the Microsoft Windows Installer (MSI). You can invoke MSI directly, without using setup.exe. This means that you can install the extended transactional function on a system without any interaction. This process is called unattended, or silent, installation.

If you are running WebSphere MQ on Windows Vista or Windows Server 2008 with User Account Control (UAC) enabled, you must invoke the silent installation from an elevated command prompt. Elevate a command prompt by using a right-click to start the command prompt and choose **Run as administrator**. If you try to silently install from a non-elevated command prompt, the install fails with an error of AMQ4353 in the install log.

To invoke an unattended installation, insert the WebSphere MQ Extended Transactional Clients CD-ROM into the CD-ROM drive and enter the following command at a command prompt:

```
msiexec /i "path\MSI\IBM WebSphere MQ Extended Transactional Client.msi" /q
TRANSFORMS="1033.mst" AGREETOLICENSE="yes"
```

where:

/q requests an unattended installation.

TRANSFORMS=″1033.mst″ specifies that the installation language is US English. For more information about installing in other national languages, see WebSphere MQ for Windows Quick Beginnings.

AGREETOLICENSE=″yes″ means that you have read the licence agreement and accept its terms.

Advanced installation methods:

WebSphere MQ for Windows is installed using the Microsoft Installer (MSI). It is possible to install WebSphere MQ by invoking MSI directly, without using setup.exe. You can use this process for more complex unattended (or silent) installation, or for interactive installation, from a command line.

Using msiexec with command line parameters:

This describes the msiexec command, its command line parameters, its property=value parameters, and its response file parameters.

You can use the msiexec command with command line parameters to invoke installation or uninstallation. At a command line, enter the following command, followed by the parameters you require:

```
msiexec
```

Table 7 on page 48 shows the parameters you can use. For a silent installation, this must include the **/q** or **/qn** parameter.

If you are running WebSphere MQ on Windows Vista or Windows Server 2008 with User Account Control (UAC) enabled, you must invoke the silent installation from an elevated command prompt. Elevate a command prompt by using a right-click to start the command prompt and choose **Run as administrator**. If you try to silently install from a non-elevated command prompt, the install fails with an error of AMQ4353 in the install log.

Note: The msiexec command can take further parameters that are not supported or listed here. If you need details of these, refer to the help file for the Windows Installer that is supplied with the MSI software development kit. See the Microsoft Web site at: http://www.microsoft.com.

A typical example of an msiexec command is:

```
msiexec /i "path\MSI\IBM WebSphere MQ.msi" /l*v c:\install.log /m mif_file
TRANSFORMS="1033.mst" ADDLOCAL="Client" REMOVE=""
```

Table 8 on page 50 and Table 9 on page 51 show the parameters that you can enter as property=value pairs on the msiexec command line (defaults are shown in bold). Note that:

- Property strings must be in upper case.
- Value strings are case-sensitive. You can enclose value strings in quotation marks. If a value string includes a blank, enclose the blank in quotation marks.
- For a property that can take more than one value, use the format: `ADDLOCAL="Client,JavaMsg,Toolkit"`

Table 7. msiexec command line parameters

Parameter	Options	Description
/a	*Package*	Installs a product on the network using administrative installation, that is, installs a source image of the application onto the network that is similar to a source image on a CD-ROM.
/i	*Package*|*ProductCode*	Installs or configures a product using the specified .msi file. The WebSphere MQ Windows Installer package is IBM WebSphere MQ.msi.
/j	[u|m]Package| [u|m]Package /t *Transform List*| [u|m]Package /g *LanguageID*	Advertises the product. This option ignores any property values entered on the command line. **u** Advertise to the current user **m** Advertise to all users of this machine **g** Language ID **t** Applies transform to advertised package
/l	[i|w|e|a|r|u|c|m|o|p |v|+|!]*Logfile*	Specifies path to log file, with flags to set which information to log. **i** Status messages **w** Recoverable warnings **e** All error messages **a** Startup of actions **r** Action-specific records **u** User requests **c** Initial user interface parameters **m** Out-of-memory or unrecoverable exit information **o** Out-of-disk-space messages **p** Terminal properties **v** Verbose output **+** Append to existing file **!** Flush each line to the log ***** Log all information except for the v option. To log all information including the v option, specify "/l*v"

Table 7. msiexec command line parameters (continued)

Parameter	Options	Description
/m	*filename*	Generates a Microsoft System Management Server (SMS) status .mif file. Must be used with either the install (/i), remove (/x), administrative installation (/a), or reinstall (/f) options. The ISMIF32.DLL is installed as part of SMS and must be on the path. The fields of the status .mif file are filled with the following information: • Manufacturer - Author • Product - Revision Number • Version - Subject • Locale - Template • Serial Number - not set • Installation - set by ISMIF32.DLL to DateTime • InstallStatus - Success or Failed • Description - Error messages in the following order: 1. Error messages generated by installer. 2. Resource from msi.dll if install could not commence or user exit 3. System error message file. 4. Formatted message: `Installer error %i`, where %i is the error returned from msi.dll.

Table 7. msiexec command line parameters (continued)

Parameter	Options	Description
/q	n|b|r|f	Sets the level of user interface displayed during the install. **q, qn** No user interface. A silent installation that displays no user interface. **qb** Basic user interface. Displays the built-in dialog boxes that show progress messages. **qr** Reduced user interface with a modal dialog box displayed at the end of the installation. **qf** Full user interface with a modal dialog box displayed at the end. **qn+** No user interface except for a modal dialog box displayed at the end of installation. **qb+** Basic user interface with a modal dialog box displayed at the end. The modal box is not displayed if the user cancels the installation. **qb-** Basic user interface with no modal dialog boxes. Note that /qb+- is not a supported UI level.
/x	*Package|ProductCode*	Uninstalls the product.

Note:
1. Do not use the options /i, /x, /j[u|m], and /a together.
2. Use the options /t and /g only with /j.
3. Use the options /l and /q with /i, /x, /j[u|m], and /a.

Table 8. msiexec property = value parameters

Property	Values	Meaning
USEINI	*path\file_name*	Use the specified response file. See "Using msiexec with a response file" on page 52.
SAVEINI	*path\file_name*	Generate a response file during installation. The file contains those parameters selected for this installation that a user might make during an interactive installation.
ONLYINI	1|yes|""	1, yes or any value other than null. End the installation before updating the target system, but after generating a response file, if this is specified. "". Continue the installation and update the target machine (the default).
TRANSFORMS	*path\file_name*	Species what transform (.mst) files should be applied to the product. For example, `"1033.mst"` specifies the supplied U.S. English transform file.

Table 9. Response file parameters

Property	Values	Meaning
PGMFOLDER	*path*	Folder for the WebSphere MQ program files. For example, `c:\mqm`.
DATFOLDER	*path*	Folder for the WebSphere MQ data files. For example, `c:\mqm\data`.
USERCHOICE	0 | no	Not used for a silent installation. If the installation is not silent, for any other value (including null), if the command line or response file specifies parameters to install features, a dialog is displayed. This dialog prompts the user to accept the preselected options, or review and possibly change them. For other types of installation, when set to 0 or no, suppresses display of the dialog.
AGREETOLICENSE	yes	Accept the terms of the license. For a silent installation, this must be set to yes. If the installation is not silent, this parameter is ignored.
ADDLOCAL	*feature*, *feature*, ... | All | ""	A comma-separated list of features to install locally.[1] All installs all features. "" installs the typical features. If you do not want a feature use REMOVE="*feature name*" **Note:** If this is a new installation the typical features (Client and Development Toolkit) are installed by default irrespective of the feature list provided in the ADDLOCAL property. If you do not want a feature, use REMOVE="*feature name*"
REMOVE	*feature*, *feature*, ... | All | ""	A comma-separated list of features to remove.[1] All uninstalls all features. "" uninstalls no features (the default).
HIGHCONTRAST	0 | no | ""	0 or no. Do not set high-contrast mode for the installation. "". (The default) Set high-contrast mode for the installation if Windows 2000 or Windows XP high-contrast mode is set or if WebSphere MQ high-contrast mode is set. Anything else. Set high-contrast mode for the installation.

1. For a list of valid feature names, see Table 6 on page 46.

Using transforms with msiexec:

This lists the local identifier, language, and the transform file name to use in the msiexec command line, to support different national languages.

MSI can use transforms to modify an installation. During WebSphere MQ installation, transforms can be used to support different national languages. WebSphere MQ is supplied with transform files in the \MSI folder of the WebSphere MQ client CD-ROM. These files are also embedded in the WebSphere MQ Windows Installer package, IBM WebSphere MQ.msi.

On the msiexec command line, you can specify the required language by using the TRANSFORMS property in a property=value pair. The quotes surrounding the value are optional. For example: `TRANSFORMS="1033.mst"`

You can also specify the full path and file name of the transform file. Again, the quotes surrounding the value are optional. For example: `TRANSFORMS="D:\Msi\1033.mst"`

Table 10 shows the locale identifier, language, and the transform file name to use in the msiexec command line. For information about the msiexec property=value parameters, see Table 8 on page 50.

You can also specify the required language by using the MQPLANGUAGE property with the MQParms command. See Table 11 on page 55.

Table 10. Supplied transform files

Language	**Transform File name**	**Locale identifier**
U.S. English	1033.mst	1033
German	1031.mst	1031
French	1036.mst	1036
Spanish	1034.mst	1034
Italian	1040.mst	1040
Brazilian Portuguese	1046.mst	1046
Japanese	1041.mst	1041
Korean	1042.mst	1042
Simplified Chinese	2052.mst	2052
Traditional Chinese	1028.mst	1028
Czech	1029.mst	1029
Russian	1049.mst	1049
Hungarian	1038.mst	1038
Polish	1045.mst	1045

Using msiexec with a response file:

You can use the msiexec command with a parameter that calls a response file, as described in "Unattended (silent) installation" on page 45. The response file contains the parameters that a user normally specifies during an interactive installation.

You can combine the msiexec command line parameters described in "Using msiexec with command line parameters" on page 47 with the response file to invoke a complex installation or uninstallation. This could be silent or interactive. For a silent installation, this must include the **/q** or **/qn** parameter.

If you are running WebSphere MQ on Windows Vista or Windows Server 2008 with User Account Control (UAC) enabled, you must invoke the silent installation from an elevated command prompt. Elevate a command prompt by using a right-click to start the command prompt and choose **Run as administrator**. If you try to silently install from a non-elevated command prompt, the install fails with an error of AMQ4353 in the install log.

To invoke the msiexec command using a response file, enter the following command at a command line:

```
msiexec [parameters] USEINI="response_file"
```

where:

parameters
: are command line parameters listed in Table 7 on page 48, or property=value pairs on the command line (always put the command line parameters first).

response_file
: is the full path and file name of the file that contains the [*Response*] stanza and the required property=value pairs, for example, C:\MyResponseFile.ini. An example response file, Response.ini, is supplied with WebSphere MQ. This file contains default installation parameters.

If a parameter is specified both on the command line and in the response file, the setting on the command line takes precedence.

For example:

```
msiexec /i "path\MSI\IBM WebSphere MQ.msi" /l*v C:\install.log /q
AGREETOLICENSE="yes" USEINI="C:\MyResponseFile.ini"
```

Using the MQParms command:

You can use the MQParms command to invoke installation or uninstallation.

This command can use parameters on a command line, or those specified in a parameter file. The parameter file is an ASCII text file that contains the parameter values that you want to set for the installation. The MQParms command takes the specified parameters and generates the corresponding msiexec command line.

This means that you can save all the parameters that you want to use with the msiexec command in a single file.

For a silent installation, this must include the **/q** or **/qn** parameter, either on the command line, or in the [MSI] stanza of the parameter file. If you are running WebSphere MQ on Windows Vista or Windows Server 2008 with User Account Control (UAC) enabled, you must invoke the silent installation from an elevated command prompt. Elevate a command prompt by using a right-click to start the command prompt and choose **Run as administrator**. If you try to silently install from a non-elevated command prompt, the install fails with an error of AMQ4353 in the install log.

You can specify many more parameters in the parameter file that you use with the MQParms command than you can in the response file that you use directly with the msiexec command.

An example of the file MQParms.ini is supplied with WebSphere MQ. This file contains default installation parameters.

There are two ways to create a parameter file for installation:

- Copy and edit the file MQParms.ini that is supplied in the root folder of the WebSphere MQ Client CD-ROM, using an ASCII file editor.
- Create your own parameter file using an ASCII file editor.

To invoke installation using the MQParms command:

1. From a command line, change to the root folder of the WebSphere MQ Client CD-ROM (that is, the location of the file MQParms.exe).
2. Enter the following command:

   ```
   MQParms [parameter_file] [parameters]
   ```

   where:

   parameter_file
   is the file that contains the required parameter values. If this file is not in the same folder as MQParms.exe, specify the full path and file name. If you do not specify a parameter file, the default is C:\MQParamsFile.ini. For further details, see "Parameter file."

   parameters
   are one or more command line parameters, as listed in Table 7 on page 48.

A typical example of an MQParms command is:

```
MQParms MyParams.ini /l*v c\:install.log
```

If you specify a parameter both on the command line and in the parameter file, the setting on the command line takes precedence.

If you do not specify /i, /x, /a, or /j, MQParms defaults to standard installation using the WebSphere MQ Windows Installer package, WebSphere MQ.msi. That is, it generates the following part of the command line: `/i` *current_folder*`\MSI\IBM WebSphere MQ.msi`

Parameter file:

A parameter file is an ASCII text file that contains sections (stanzas) with parameters that can be used by the MQParms command. Typically, this is an initialization file such as MQParms.ini.

The MQParms command takes parameters from the following stanza in the file:

[MSI] Contains general properties related to how the MQParms command runs and to the installation of WebSphere MQ.

The properties that you can set in this stanza are listed in Table 7 on page 48, Table 8 on page 50, Table 9 on page 51, and Table 11 on page 55.

MQParms ignores any other stanzas in the file.

In the [MSI] stanza, the properties can be in command line format (for example, **/q**) or property=value format.

Some properties can take more than one value, for example:
|
```
ADDLOCAL="Client,JavaMsg,Toolkit"
```

To clear a property, set its value to an empty string, for example: `REMOVE=""`

You can enter parameters in command line format (for example, **/q**) and in property=value format (for example, `ADDLOCAL="Client"`). Refer to Table 9 on page 51, Table 7 on page 48, and Table 8 on page 50 for the properties used to install WebSphere MQ.

Table 11 shows additional properties in the stanza that affect how the MQParms command runs, but that do not affect the installation.

A typical example of a parameter file is:

```
[MSI] MQPLANGUAGE=1033 MQPLOG=%temp%\MQParms.log MQPSMS=1 ADDLOCAL=Client
/m miffile REMOVE="" /l*v c:\install.log
```

Table 11. Properties used by MQParms in the MSI stanza

Property	Values	Description
MQPLOG	*path*|*file_name*	MQParms generates a text log file with the specified name and location.
MQPLANGUAGE	**system**|user| *transform_value*	The installation language. system. Install using the language of the default system locale (the default). user. Install using the language of the default locale of the user. *transform_value*. Install using the language specified by this value. See Table 10 on page 52.
MQPSMS	**0**|no	0 or no. MQParms does not wait for the msiexec command to end (the default). Any other value. MQParms waits for the msiexec command to end.
MQPINUSE	**0**|1	If MQPINUSE is set to 1, MQParams continues installing even if WebSphere MQ files are in use. If this option is used a reboot will be required to complete the installation.
MQPNOREBOOT	**0**|1	If MQPNOREBOOT is set to 1, the reboot that is required if installation takes place while WebSphere MQ files are still in use will be suppressed.

Using Microsoft System Management Server:

WebSphere MQ is supplied with two sample definition files to create a System Management Server (SMS) Package (these can be found on the WebSphere MQ server CD).

These are:
- WebSphere MQ.pdf
- WebSphere MQ.sms

You will need to update the **CommandLine** parameter supplied in the definition files by stating the path to where you have the file **IBM WebSphere MQ.msi**. This file is supplied on the WebSphere MQ server CD by going to MSI\IBM WebSphere MQ.msi.

Please refer to the Microsoft System Management Server documentation for the version of SMS you are using to get full information on how to create and run a job.

Once the Package has been created, an SMS job can be configured to install WebSphere MQ.

Note: You must be logged onto the target machine with Administrator authority to install WebSphere MQ.

Installing the extended transactional client using SMS:

To install the extended transactional client, proceed as above, but using the WebSphere MQ Extended Transactional Clients CD.

Modifying a WebSphere MQ client installation silently

Silently modifying a WebSphere MQ client installation using msiexec:

To silently modify an installation using msiexec, follow the instructions on the installation pages, but set the ADDLOCAL parameter to include the features you want to add, and set the REMOVE parameter to the features you want to remove.

For example if you used ADDLOCAL=″JavaMsg″ and REMOVE=″″ it would modify the installation to include the Java Messaging and Web Services feature.

The instructions for msiexec begin here: “Using msiexec with command line parameters” on page 47

Silently modifying a WebSphere MQ client installation using MQParms:

To silently modify an installation using MQParms, follow the instructions on the installation pages, but set the ADDLOCAL parameter to include the features you want to add, and set the REMOVE parameter to the features you want to remove.

For example if you used ADDLOCAL=″JavaMsg″ and REMOVE=″″ it would modify the installation to include the Java Messaging and Web Services feature.

The instructions for MQParms begin here: “Using the MQParms command” on page 53

Uninstalling a WebSphere MQ client

This section describes how to uninstall (remove) a WebSphere MQ client if you installed it using the WebSphere MQ Client CD.

You can uninstall (remove) WebSphere MQ client in attended mode or unattended (silent) mode.

Before you uninstall WebSphere MQ client, ensure that there are no WebSphere MQ client programs running.

Uninstalling WebSphere MQ client from Windows:

There are three ways to uninstall WebSphere MQ from your machine:

- Start the installation process, then select the appropriate option.
- On Windows 2003 and Windows XP, use the Add/Remove Programs facility in the Windows Control Panel. On Windows Vista and Windows Server 2008, use the **Uninstall** button in the Programs and Features facility in the Windows Control Panel.
- Perform a removal from the command line.

You can use these methods to uninstall the WebSphere MQ client, provided that the original installation used the WebSphere MQ Client CD.

You can also uninstall WebSphere MQ client by using the appropriate parameters with advanced installation methods, or by using Microsoft System Management Server (SMS). See "Advanced installation methods" on page 47.

Uninstalling WebSphere MQ client using the installation process:

This procedure uninstalls WebSphere MQ from your machine in attended mode. It removes all the currently installed features.

1. Insert WebSphere MQ Client CD 5 into the CD drive.
2. If autorun is installed, the installation process starts.

   Otherwise, double-click the **Setup** icon in the root folder of the CD to start the installation process.

   The WebSphere MQ Client Setup window is displayed. Click **Next** to continue.
3. Select **Remove**, then click **Next**.
4. The Remove WebSphere MQ panel is displayed, with a summary of the installation to be removed.

   Click **Remove** to continue.
5. The Removing WebSphere MQ panel is displayed.

   Wait for the progress bar to complete.

   If there are any messages that state that locked files are found, ensure that there are no WebSphere MQ client programs running.

   Uninstallation should then continue.
6. The WebSphere MQ Setup window displays the following message:

   Uninstallation Completed Successfully

   Click **Finish**.

Uninstalling WebSphere MQ client using Add/Remove Programs or Programs and Features:

For Windows 2000, Windows XP, and Windows 2003, follow these steps:

1. From the Windows taskbar, select **Start→ Settings→ Control Panel**. The Add/Remove Programs window opens.
2. Double-click **Add/Remove Programs**.
3. Select **IBM WebSphere MQ**.
4. Do one of the following:
   - Select **Remove**. When a confirmation prompt is displayed, select **Yes**.

     The uninstall program begins. All the WebSphere MQ files are removed.

- Select **Change**. The WebSphere MQ Setup window with the Program Maintenance panel is displayed. Follow the procedure for uninstalling WebSphere MQ using the process from step 3 on page 57 to the end.

For Windows Vista and Windows Server 2008, follow these steps:

1. From the Windows taskbar, click **Start** → **Control Panel** → **Programs**. The Programs window opens.
2. Double-click **Programs and Features**. The Programs and Features window opens.
3. Select **IBM WebSphere MQ**.
4. Click the **Uninstall** button. A window containing a confirmation prompt opens. Click **Yes**. If UAC is enabled, accept the Windows prompt to allow the uninstall to run as elevated. The uninstall program then begins and runs to completion. All the WebSphere MQ files are removed.

Uninstalling WebSphere MQ client using the command line:

This procedure can be used for removing the WebSphere MQ files in unattended (silent) mode.

To invoke an uninstallation, you use the `msiexec` command.

To uninstall all WebSphere MQ client features, enter one of the following commands:

- `msiexec /i "path\MSI\IBM WebSphere MQ.msi" REMOVE="All"`

  This command gives you an interactive uninstallation of all features.

  If you are running WebSphere MQ on Windows Vista or Windows Server 2008 with User Account Control (UAC) enabled, you might see Open File - Security Warning dialog boxes that list International Business Machines Limited as the publisher. Click **Run** to allow the uninstallation to continue.
- `msiexec /i "path\MSI\IBM WebSphere MQ.msi" /q REMOVE="All"`

  This command invokes a silent uninstall of all features.

  If you are running WebSphere MQ on Windows Vista or Windows Server 2008 with User Account Control (UAC) enabled, you must invoke the silent uninstallation from an elevated command prompt.
- `msiexec /x "path\MSI\IBM WebSphere MQ.msi"`

  This command displays only a progress dialog whilst uninstalling all features.

  If you are running WebSphere MQ on Windows Vista or Windows Server 2008 with User Account Control (UAC) enabled, you might see Open File - Security Warning dialog boxes that list International Business Machines Limited as the publisher. Click **Run** to allow the uninstallation to continue.
- `msiexec /x "path\MSI\IBM WebSphere MQ.msi" /q`

  This command invokes a silent uninstall of all features.

  If you are running WebSphere MQ on Windows Vista or Windows Server 2008 with User Account Control (UAC) enabled, you must invoke the silent uninstallation from an elevated command prompt.

Alternatively, you can use the `msiexec` command with a parameter that calls a response file. A response file is an ASCII text file that contains the parameter values you want to set for the uninstallation. The response file has a format similar to a Windows .ini file, and contains the stanza **[Response]**. This stanza contains parameters that the `msiexec` command can use, in the form of PROPERTY=value pairs. The `msiexec` command ignores any other stanzas in the file.

You can set which features to uninstall.

Note: The response file you use to uninstall WebSphere MQ for Windows when it was installed using WebSphere MQ Client CD is *not* the same as the one used with earlier non-MSI versions of WebSphere MQ. For details about the response file you use with WebSphere MQ Client CD, see "Unattended (silent) installation" on page 45.

To uninstall WebSphere MQ using a response file, enter the following command:

```
msiexec /i "path\MSI\IBM WebSphere MQ.msi" /q USEINI="response_file"
```

where *response_file* is the file that contains the [Response] stanza and the required PROPERTY=value pairs.

An example of a typical uninstallation response file is:

```
[Response]
REMOVE="Client,JavaMsg,Toolkit"
```

Uninstalling the extended transactional function:

This section describes how to uninstall, or remove, the extended transactional function, leaving the WebSphere MQ base client on your system.

For information about how to uninstall the WebSphere MQ base client after you have uninstalled the extended transactional function, see "Uninstalling WebSphere MQ client from Windows" on page 56.

You can uninstall the extended transactional function in the following ways:

- Start the installation process from the WebSphere MQ Server CD. This gives you the option to uninstall the extended transactional client.
- On Windows XP and Windows 2003: use **Add/Remove Programs** in the Control Panel for Windows.
- Uninstall from a command prompt.

You can also uninstall the extended transactional function by using SMS.

Before you uninstall the extended transactional function, ensure that there are no WebSphere MQ client applications running.

Uninstalling the extended transactional function using the installation process:

This procedure uninstalls the extended transactional function from your system, and applies to all versions of Windows

1. Insert WebSphere MQ Server CD into the CD drive.
2. If autorun is installed, the installation process starts.

   Otherwise, double-click on the Setup icon in the root folder of the CD to start the installation process.

   The WebSphere MQ Client Setup window is displayed. Click **Next** to continue.
3. Select **Modify**, then click **Next**.

   The Features panel is displayed.
4. Select to uninstall the **Extended Transactional Client feature**:

   a. Click on the symbol to the left of **Client Extended Transaction Support** to display a drop-down menu.
   b. Select the **Do not install this feature (remove if already installed).** option. The symbol to the left of the feature name changes to show the current installation option.
5. When your selections are complete, click **Next**.
6. The WebSphere MQ Client Setup window displays a summary of the installation you selected.

   To continue, click **Modify**.
7. Wait until the progress bar is complete.

   When the WebSphere MQ client is successfully installed, the WebSphere MQ Client Setup window displays the following message:

   Installation Wizard Completed Successfully

   Click **Finish** to close the window.

Uninstalling the extended transactional function using Add/Remove Programs:

This procedure uninstalls the extended transactional function from your system, and only applies to Windows XP and Windows 2003:
1. From the Windows taskbar, click **Start —> Settings —> Control Panel**. The Control Panel window opens.
2. Double-click **Add/Remove Programs**. The Add/Remove Programs window opens.
3. Click **IBM WebSphere MQ** to select it.
4. Click **Change**. The WebSphere MQ V7.0 Setup window will open.
5. Click **Next**. Select the radio button next to **Modify** and then click **Next**.
6. Click the symbol to the left of **Client Extended Transactional Support** to display a drop-down menu.
7. Select **Do not install this feature (remove if already installed)**. Then click **Next**
8. Click **Modify** to continue.
9. Wait until the progress bar is complete.

   When the WebSphere MQ client is successfully installed, the WebSphere MQ Client Setup window displays the following message:

   Installation Wizard Completed Successfully

   Click **Finish** to close the window.

Uninstalling the extended transactional function from a command prompt:

This method can be used for uninstalling the extended transactional function in unattended mode.

The method uses the msiexec command.

To uninstall the extended transactional function, insert the WebSphere MQ Server CD-ROM into the CD-ROM drive and enter one of the following commands at a command prompt:
- `msiexec /i "path\MSI\IBM WebSphere MQ.msi" REMOVE="XA_Client"`

  This command gives you an interactive uninstallation of the extended transactional function.

If you are running WebSphere MQ on Windows Vista or Windows Server 2008 with User Account Control (UAC) enabled, you might see Open File - Security Warning dialog boxes that list International Business Machines Limited as the publisher. Click **Run** to allow the uninstallation to continue.

- `msiexec /i "path\MSI\IBM WebSphere MQ.msi" /q REMOVE="XA_Client"`

  This command invokes a silent uninstall of the extended transactional function.

  If you are running WebSphere MQ on Windows Vista or Windows Server 2008 with User Account Control (UAC) enabled, you must invoke the silent uninstallation from an elevated command prompt.
- `msiexec /x "path\MSI\IBM WebSphere MQ.msi"`

  This command displays a progress dialog whilst uninstalling all features, including the extended transactional function.

  If you are running WebSphere MQ on Windows Vista or Windows Server 2008 with User Account Control (UAC) enabled, you might see Open File - Security Warning dialog boxes that list International Business Machines Limited as the publisher. Click Run to allow the uninstallation to continue.
- `msiexec /x "path\MSI\IBM WebSphere MQ.msi" /q`

  This command invokes a silent uninstall of all features, including the extended transactional function.

  If you are running WebSphere MQ on Windows Vista or Windows Server 2008 with User Account Control (UAC) enabled, you must invoke the silent uninstallation from an elevated command prompt.

Silently uninstalling a WebSphere MQ client installation using MQParms:

To silently uninstall using MQParms, follow the instructions on the installation pages, but set the ADDLOCAL parameter to empty, and set the REMOVE parameter to ″ALL″.

For example ADDLOCAL=″″ and REMOVE=″ALL″.

The instructions for MQParms begin here: “Using the MQParms command” on page 53

Configuring communication links

This chapter provides an overview of configuring the WebSphere MQ client and server communication links, and how to enable the server to listen for communications from the WebSphere MQ client.

In WebSphere MQ, the logical communication links between objects are called *channels*. The channels used to connect WebSphere MQ clients to servers are called MQI channels. You set up channel definitions at each end of your link so that your WebSphere MQ application on the WebSphere MQ client can communicate with the queue manager on the server. There is a detailed description of how to do this in “Using channels” on page 86.

Before you define your MQI channels, you need to:

1. Decide on the form of communication you are going to use. See “Deciding which communication type to use” on page 62
2. Define the connection at each end of the channel:

   To define the connection, you need to:

   - Configure the connection.

- Record the values of the parameters that you need for the channel definitions.
- Enable the server to detect incoming network requests from your WebSphere MQ client. This involves starting a *listener*.

Deciding which communication type to use

There are four types of communication for MQI channels on different platforms:

- LU 6.2
- NetBIOS
- SPX
- TCP/IP

When you define your MQI channels, each channel definition must specify a transmission protocol (transport type) attribute. A server is not restricted to one protocol, so different channel definitions can specify different protocols. For WebSphere MQ clients, it might be useful to have alternative MQI channels using different transmission protocols.

Your choice of transmission protocol also depends on your particular combination of WebSphere MQ client and server platforms. The possible combinations are shown in the following table.

Table 12. Transmission protocols - combination of WebSphere MQ client and server platforms

Transmission protocol	**WebSphere MQ client**	**WebSphere MQ server**
TCP/IP	UNIX systems Windows	i5/OS UNIX systems Windows z/OS
LU 6.2	UNIX systems[1] Windows	i5/OS UNIX systems[1] Windows z/OS
NetBIOS	Windows	Windows
SPX	Windows	Windows
Note: 1. Except Linux (POWER platform)		

Defining a connection is described in the following sections:

- "Defining a TCP/IP connection" on page 63
- "Defining an LU 6.2 connection" on page 64
- "Defining a NetBIOS connection" on page 64
- "Defining an SPX connection" on page 64

Defining a TCP/IP connection

Defining a TCP/IP connection on a WebSphere MQ client

You specify a TCP/IP connection at the client by specifying a transport type of TCP on the channel definition.

See the section *Defining a TCP connection* for your client platform in WebSphere MQ Intercommunication.

Defining a TCP/IP connection on a WebSphere MQ server

Receiving channel programs are started in response to a startup request from the sending channel. To do this, a listener program has to be started to detect incoming network requests and start the associated channel. The procedure for starting a listener program depends on the server platform. See the section *Defining a TCP connection* for your server platform in WebSphere MQ Intercommunication.

TCP/IP connection limits

On any platform, there is a limit to the number of outstanding connection requests that can be queued at a single TCP/IP port.

This is not the same as the maximum number of clients you can attach to a WebSphere MQ server. You can connect more clients to a server, up to the level determined by the server system resources. The backlog values for connection requests are shown in the following table:

Table 13. Maximum outstanding connection requests queued at a TCP/IP port

Server platform	**Maximum connection requests**
AIX	100
HP-UX	20
Linux	100
i5/OS	255
Solaris	100
Windows Server	100
Windows Workstation	100
z/OS	255

If the connection limit is reached, the client receives a return code of
| MQRC_HOST_NOT_AVAILABLE from the **MQCONN** call, and an AMQ9202 error in the client error log (var/mqm/AMQERR0n on UNIX systems or amqerr0n.log in the errors subdirectory of the WebSphere MQ client installation on Windows). If the client retries the **MQCONN** request, it might be successful.

To increase the number of connection requests you can make, and avoid error messages being generated by this limitation, you can have a listener listening on more than one port, or have more than one queue manager.

Defining an LU 6.2 connection

Defining an LU 6.2 connection on a WebSphere MQ client

The method of defining an LU 6.2 connection varies according to the client platform you are using. See the section *Defining an LU 6.2 connection* for your client platform in WebSphere MQ Intercommunication.

Defining an LU 6.2 Connection on a WebSphere MQ server

The method of defining an LU 6.2 connection varies according to the server platform you are using. See the section *Defining an LU 6.2 connection* for your server platform in WebSphere MQ Intercommunication.

Defining a NetBIOS connection

A NetBIOS connection applies only to a client and server running Windows. See the section *Defining a NetBIOS connection* for Windows in WebSphere MQ Intercommunication.

Defining an SPX connection

An SPX connection applies only to a client and server running Windows XP and Windows 2003 Server. See the section *Defining an SPX connection* for Windows in WebSphere MQ Intercommunication.

Configuring an extended transactional client

For each platform, the extended transactional client provides support for the following external transaction managers:

XA compliant transaction managers
The extended transactional client provides the XA resource manager interface to support XA compliant transaction managers such as CICS® and Tuxedo.

Microsoft Transaction Server (Windows systems only)
On Windows systems only, the XA resource manager interface also supports Microsoft Transaction Server (MTS). The WebSphere MQ MTS support supplied with the extended transactional client provides the bridge between MTS and the XA resource manager interface.

WebSphere Application Server
The extended transactional client supports WebSphere MQ JMS applications that connect to a queue manager in client mode and use WebSphere Application Server as the transaction manager.

This chapter describes how to configure the extended transactional function for each category of transaction manager. The chapter contains the following sections:

- "XA compliant transaction managers" on page 65
- "Microsoft Transaction Server" on page 73
- "WebSphere Application Server" on page 74

XA compliant transaction managers

Note: This section assumes that you have a basic understanding of the XA interface as published by The Open Group in *Distributed Transaction Processing: The XA Specification*.

To configure an extended transactional client, you must first configure the WebSphere MQ base client as described in "Installing client components" on page 22. Using the information in this section, you can then configure the extended transactional function for an XA compliant transaction manager such as CICS and Tuxedo.

A transaction manager communicates with a queue manager as a resource manager using the same MQI channel as that used by the client application that is connected to the queue manager. When the transaction manager issues a resource manager (xa_) function call, the MQI channel is used to forward the call to the queue manager, and to receive the output back from the queue manager.

Either the transaction manager can start the MQI channel by issuing an xa_open call to open the queue manager as a resource manager, or the client application can start the MQI channel by issuing an MQCONN or MQCONNX call.

- If the transaction manager starts the MQI channel, and the client application calls MQCONN or MQCONNX subsequently on the same thread, the MQCONN or MQCONNX call completes successfully and a connection handle is returned to the application. The application does not receive a MQCC_WARNING completion code with an MQRC_ALREADY_CONNECTED reason code.
- If the client application starts the MQI channel, and the transaction manager calls xa_open subsequently on the same thread, the xa_open call is forwarded to the queue manager using the MQI channel.

In a recovery situation following a failure, when no client applications are running, the transaction manager can use a dedicated MQI channel to recover any incomplete units of work in which the queue manager was participating at the time of the failure.

Note the following conditions when using an extended transactional client with an XA compliant transaction manager:

- Within a single thread, a client application can be connected to only one queue manager at a time. Note that this restriction applies only when using an extended transactional client; a client application that is using a WebSphere MQ base client can be connected to more than one queue manager concurrently within a single thread.
- Each thread of a client application can connect to a different queue manager.
- A client application cannot use shared connection handles.

To configure the extended transactional function, you must provide the following information to the transaction manager for each queue manager that acts as a resource manager:

- An xa_open string
- A pointer to an XA switch structure

When the transaction manager calls xa_open to open the queue manager as a resource manager, it passes the xa_open string to the extended transactional client as the argument, *xa_info*, on the call. The extended transactional client uses the information in the xa_open string in the following ways:

- To start an MQI channel to the server queue manager, if the client application has not already started one
- To check that the queue manager that the transaction manager opens as a resource manager is the same as the queue manager to which the client application connects
- To locate the transaction manager's ax_reg and ax_unreg functions, if the queue manager uses dynamic registration

For the format of an xa_open string, and for more details about how the information in the xa_open string is used by an extended transactional client, see "The xa_open string."

An XA switch structure enables the transaction manager to locate the xa_ functions provided by the extended transactional client, and specifies whether the queue manager uses dynamic registration. For information about the XA switch structures supplied with an extended transactional client, see "The XA switch structures" on page 70.

For information about how to configure the extended transactional function for a particular transaction manager, and for any other information about using the transaction manager with an extended transactional client, see the following sections:

- "Configuring for CICS" on page 71
- "Configuring for Tuxedo" on page 72

The xa_open string

This section describes the format of an xa_open string and provides more details about how an extended transactional client uses the information in an xa_open string.

The format of an xa_open string:

An xa_open string has the following format:

```
parm_name1=parm_value1,parm_name2=parm_value2, ...
```

where `parm_name` is the name of a parameter and `parm_value` is the value of a parameter. The names of the parameters are not case sensitive but, unless stated otherwise subsequently, the values of the parameters are case sensitive. You can specify the parameters in any order.

The names, meanings, and valid values of the parameters are as follows:

Name **Meaning and valid values**

CHANNEL
The name of an MQI channel.

This is an optional parameter. If this parameter is supplied, the CONNAME parameter must also be supplied.

TRPTYPE
The communications protocol for the MQI channel. The following are valid values:

LU62 SNA LU 6.2

NETBIOS
NetBIOS

SPX IPX/SPX

TCP TCP/IP

This is an optional parameter. If it is omitted, the default value of TCP is assumed. The values of the parameter are not case sensitive.

CONNAME
The network address of the queue manager at the server end of the MQI channel. The valid values of this parameter depend on the value of the TRPTYPE parameter:

LU62 A symbolic destination name, which identifies a CPI-C side information entry.

Note that the network qualified name of a partner LU is not a valid value, nor is a partner LU alias. This is because there are no additional parameters to specify a transaction program (TP) name and a mode name.

NETBIOS
A NetBIOS name.

SPX A 4-byte network address, a 6-byte node address, and an optional 2-byte socket number. These values must be specified in hexadecimal notation. A period must separate the network and node addresses, and the socket number, if supplied, must be enclosed in parentheses. For example:

```
0a0b0c0d.804abcde23a1(5e86)
```

If the socket number is omitted, the default value of 5e86 is assumed.

TCP A host name or an IP address, optionally followed by a port number in parentheses. If the port number is omitted, the default value of 1414 is assumed.

This is an optional parameter. If this parameter is supplied, the CHANNEL parameter must also be supplied.

QMNAME
The name of the queue manager at the server end of the MQI channel. The name cannot be blank or a single asterisk (*), nor can the name start with an asterisk. This means that the parameter must identify a specific queue manager by name.

This is a mandatory parameter.

TPM The transaction manager being used. The valid values are CICS and TUXEDO.

An extended transactional client uses this parameter and the AXLIB parameter for the same purpose. For more information these parameters, see "The TPM and AXLIB parameters" on page 68.

This is an optional parameter. The values of the parameter are not case sensitive.

AXLIB
The name of the library that contains the transaction manager's ax_reg and ax_unreg functions.

This is an optional parameter.

Here is an example of an xa_open string:

```
channel=MARS.SVR,trptype=tcp,conname=MARS(1415),qmname=MARS,tpm=cics
```

How an extended transactional client uses an xa_open string:

The following sections describe how an extended transactional client uses the parameters in an xa_open string.

The CHANNEL, TRPTYPE, CONNAME, and QMNAME parameters:

If the CHANNEL and CONNAME parameters are supplied in the xa_open string, the extended transactional client uses these parameters, and the TRPTYPE parameter, to start an MQI channel to the server queue manager.

If the CHANNEL and CONNAME parameters are not supplied in the xa_open string, the extended transactional client uses the value of the MQSERVER environment variable to start an MQI channel. If the MQSERVER environment variable is not defined, the extended transactional client uses the entry in the client channel definition identified by the QMNAME parameter.

In each of these cases, the extended transactional client checks that the value of the QMNAME parameter is the name of the queue manager at the server end of the MQI channel. If it is not, the xa_open call fails and the transaction manager reports the failure to the application.

When the client application calls MQCONN or MQCONNX subsequently on the same thread that the transaction manager used to issue the xa_open call, the application receives a connection handle for the MQI channel that was started by the xa_open call. A second MQI channel is not started. The extended transactional client checks that the value of the *QMgrName* parameter on the MQCONN or MQCONNX call is the name of the queue manager at the server end of the MQI channel. If it is not, the MQCONN or MQCONNX call fails with a reason code of MQRC_ANOTHER_Q_MGR_CONNECTED. If the value of the *QMgrName* parameter is blank or a single asterisk (*), or starts with an asterisk, the MQCONN or MQCONNX call fails with a reason code of MQRC_Q_MGR_NAME_ERROR.

If the client application has already started an MQI channel by calling MQCONN or MQCONNX before the transaction manager calls xa_open on the same thread, the transaction manager uses this MQI channel instead. A second MQI channel is not started. The extended transactional client checks that the value of the QMNAME parameter in the xa_open string is the name of the server queue manager. If it is not, the xa_open call fails.

If a client application starts an MQI channel first, the value of the *QMgrName* parameter on the MQCONN or MQCONNX call can be blank or a single asterisk (*), or it can start with an asterisk. Under these circumstances, however, you must ensure that the queue manager to which the application connects is the same as the queue manager that the transaction manager intends to open as a resource manager when it calls xa_open subsequently on the same thread. You might encounter fewer problems, therefore, if the value of the *QMgrName* parameter identifies the queue manager explicitly by name.

The TPM and AXLIB parameters:

An extended transactional client uses the TPM and AXLIB parameters to locate the transaction manager's ax_reg and ax_unreg functions. These functions are used only if the queue manager uses dynamic registration.

If the TPM parameter is supplied in an xa_open string, but the AXLIB parameter is not supplied, the extended transactional client assumes a value for the AXLIB parameter based on the value of the TPM parameter. See Table 14 for the assumed values of the AXLIB parameter.

Table 14. Assumed values of the AXLIB parameter

Value of TPM	Platform	Assumed value of AXLIB
CICS	AIX	/usr/lpp/encina/lib/libEncServer.a(EncServer_shr.o)
CICS	HP-UX	/opt/encina/lib/libEncServer.sl
CICS	Solaris	/opt/encina/lib/libEncServer.so
CICS	Windows systems	libEncServer
Tuxedo	AIX	/usr/lpp/tuxedo/lib/libtux.a(libtux.so.60)
Tuxedo	HP-UX	/opt/tuxedo/lib/libtux.sl
Tuxedo	Solaris	/opt/tuxedo/lib/libtux.so.60
Tuxedo	Windows systems	libtux

If the AXLIB parameter is supplied in an xa_open string, the extended transactional client uses its value to override any assumed value based on the value of the TPM parameter. The AXLIB parameter can also be used for a transaction manager for which the TPM parameter does not have a specified value.

Additional error processing:

The xa_open call also fails if any of the following occur:
- There are errors in the xa_open string.
- There is insufficient information to start an MQI channel.
- There is a problem while trying to start an MQI channel (the server queue manager is not running, for example).

Recovery processing:

Following a failure, a transaction manager must be able to recover any incomplete units of work. To do this, the transaction manager must be able to open as a resource manager any queue manager that was participating in an incomplete unit of work at the time of the failure.

If you ever need to change any configuration information, therefore, you must ensure that all incomplete units of work have been resolved before making the changes. Alternatively, you must ensure that the configuration changes do not affect the transaction manager's ability to open the queue managers it needs to open. The following are examples of such configuration changes:
- Changing the contents of an xa_open string
- Changing the value of the MQSERVER environment variable
- Changing entries in the client channel definition table
- Deleting a server connection channel definition

The XA switch structures

Two XA switch structures are supplied with the extended transactional client on each platform:

MQRMIXASwitch
: This switch structure is used by a transaction manager when a queue manager, acting as a resource manager, is not using dynamic registration.

MQRMIXASwitchDynamic
: This switch structure is used by a transaction manager when a queue manager, acting as a resource manager, uses dynamic registration.

These switch structures are located in the libraries shown in Table 15.

Table 15. WebSphere MQ libraries containing the XA switch structures

Platform	Library containing the XA switch structures
AIX	/usr/mqm/lib/libmqcxa
HP-UX Linux Solaris	/opt/mqm/lib/libmqcxa
Windows systems	C:\Program Files\IBM\WebSphere MQ\bin\mqcxa.dll [1]

Note:
1. This is the default location for the library, but you might have installed it in a different location.

The name of the WebSphere MQ resource manager in each switch structure is MQSeries_XA_RMI, but many queue managers can share the same switch structure.

Dynamic registration:

If a queue manager does not use dynamic registration, a transaction manager involves the queue manager in every unit of work. The transaction manager does this by calling xa_start, xa_end, and xa_prepare, even if the queue manager has no resources that are updated within the unit of work.

If a queue manager uses dynamic registration, a transaction manager starts by assuming that the queue manager is not involved a unit or work, and does not call xa_start. The queue manager then becomes involved in the unit of work only if its resources are updated within syncpoint control. If this occurs, the extended transactional client calls ax_reg to register the queue manager's involvement.

Using dynamic registration is a form of optimization because it can reduce the number of xa_ function calls issued by the transaction manager.

Using the Extended transactional client with SSL channels

You cannot set up an SSL channel using the xa_open string. Follow these instructions to use the client channel definition table.

About this task

Because of the limited size of the xa_open xa_info string, it is not possible to pass all the information required to set up an SSL channel using the xa_open string method of connecting to a queue manager. Therefore you must either use the client channel definition table or, if your transaction manager allows, create the channel with MQCONNX before issuing the xa_open call.

To use the client channel definition table, follow these steps:
1. Specify an xa_open string containing only the mandatory qmname (queue manager name) parameter, for example: `XA_Open_String=qmname=MYQM`
2. Use a queue manager to define a CLNTCONN (client-connection) channel with the required SSL parameters. Include the queue manager name in the QMNAME attribute on the CLNTCONN definition. This will be matched up with the qmname in the xa_open string.
3. Make the CLNTCONN definition available to the client system in a client channel definition table (CCDT) or, on Windows, in the active directory.
4. If you are using a CCDT, identify the CCDT containing the definition of the CLNTCONN channel using environment variables MQCHLLIB and MQCHLTAB. Set these variables in the environments used by both the client application and the transaction manager.

Results

This gives the transaction manager a channel definition to the appropriate queue manager with the SSL attributes needed to authenticate correctly, including SSLCIPH, the CipherSpec.

Configuring for CICS

You configure an extended transactional client for use by CICS by adding an XAD resource definition to a CICS region.

You can do this by using the CICS resource definition online (RDO) command, **cicsadd**. The XAD resource definition specifies the following information:
- An xa_open string
- The fully qualified path name of a switch load file

One switch load file is supplied for use by CICS on each of the following platforms: AIX, HP-UX, Solaris, and Windows systems. Each switch load file contains a function that returns a pointer to the XA switch structure that is used for dynamic registration, MQRMIXASwitchDynamic. See Table 16 for the fully qualified path name of each switch load file.

Table 16. The switch load files

Platform	Switch load file
AIX	/usr/mqm/lib/amqczsc
HP-UX Solaris	/opt/mqm/lib/amqczsc
Windows systems	C:\Program Files\IBM\WebSphere MQ\bin\mqcc4swi.dll [1]
Note: 1. This is the default location for the file, but you might have installed it in a different location.	

Note that you cannot use the switch load files with TXSeries Version 5.0. However, the source of the switch load files is provided in amqzscix.c, for AIX, HP-UX, and

Solaris, and in amqzscin.c, for Windows systems. You can therefore build your own switch load file to use with TXSeries Version 5.0. You might also build your own switch load file if, for example, you do not want to use dynamic registration.

Here is an example of an XAD resource definition for Windows systems:

```
cicsadd -c xad -r REGION1 WMQXA \
    ResourceDescription="WebSphere MQ queue manager MARS" \
    XAOpen="channel=MARS.SVR,trptype=tcp,conname=MARS(1415),qmname=MARS,tpm=cics" \
    SwitchLoadFile="C:\Program Files\IBM\WebSphere MQ\bin\mqcc4swi.dll"
```

For more information about adding an XAD resource definition to a CICS region, see the *CICS Administration Reference* and the *CICS Administration Guide* for your platform.

Note the following information about using CICS with an extended transactional client:

- You can add only one XAD resource definition for WebSphere MQ to a CICS region. This means that only one queue manager can be associated with a region, and all CICS applications that run in the region can connect only to that queue manager. If you want to run CICS applications that connect to a different queue manager, you must run the applications in a different region.
- Each application server in a region calls xa_open while it is initializing and starts an MQI channel to the queue manager associated with the region. This means that the queue manager must be started before an application server starts, otherwise the xa_open call fails. All WebSphere MQ client applications processed by the application server subsequently use the same MQI channel.
- When an MQI channel starts, and there is no security exit at the client end of the channel, the user ID that flows from the client system to the server connection MCA is `cics`. Under certain circumstances, the queue manager uses this user ID for authority checks when the server connection MCA subsequently attempts to access the queue manager's resources on behalf of a client application. If this user ID is used for authority checks, you must ensure that it has the authority to access all the resources it needs to access.

  For information about when the queue manager uses this user ID for authority checks, see WebSphere MQ Security.
- The CICS task termination exits that are supplied for use on WebSphere MQ client systems are listed in Table 17. You configure these exits in the same way that you configure the corresponding exits for WebSphere MQ server systems. For this information, therefore, see the WebSphere MQ System Administration Guide.

Table 17. CICS task termination exits

Platform	Source	Library
AIX HP-UX Solaris	amqzscgx.c	amqczscg
Windows systems	amqzscgn.c	mqcc1415.dll

Configuring for Tuxedo

To configure an extended transactional client for use by Tuxedo, do the following:

- In the GROUPS section of the Tuxedo UBBCONFIG file for an application, use the OPENINFO parameter to specify an xa_open string.

For an example of how to do this, see the sample UBBCONFIG file, which is supplied for use with the Tuxedo sample programs. On AIX, HP-UX, and Solaris, the name of the file is ubbstxcx.cfg and, on Windows systems, the name of the file is ubbstxcn.cfg .

- In the entry for a queue manager in the Tuxedo resource manager table:
  - udataobj/RM (AIX, HP-UX, and Solaris)
  - udataobj\rm (Windows systems)

  specify the name of an XA switch structure and the fully qualified path name of the library that contains the structure. For an example of how to do this for each platform, see the section that describes the Tuxedo sample programs in the WebSphere MQ Application Programming Guide. Tuxedo supports dynamic registration of a resource manager, and so you can use either MQRMIXASwitch or MQRMIXASwitchDynamic.

Microsoft Transaction Server

No additional configuration is required before you can use MTS as a transaction manager.

Note the following information about using MTS with the extended transactional client:

- An MTS application always starts an MQI channel when it connects to a server queue manager. MTS, in its role as a transaction manager, then uses the same MQI channel to communicate with the queue manager.
- Following a failure, MTS must be able to recover any incomplete units of work. To do this, MTS must be able to communicate with any queue manager that was participating in an incomplete unit of work at the time of the failure.

  When an MTS application connects to a server queue manager and starts an MQI channel, the extended transactional client extracts sufficient information from the parameters of the MQCONN or MQCONNX call to enable the channel to be restarted following a failure, if required. The extended transactional client passes the information to MTS, and MTS records the information in its log.

  If the MTS application issues an MQCONN call, this information is simply the name of the queue manager. If the MTS application issues an MQCONNX call and provides a channel definition structure, MQCD, the information also includes the name of the MQI channel, the network address of the server queue manager, and the communications protocol for the channel.

  In a recovery situation, MTS passes this information back to the extended transactional client, and the extended transactional client uses it to restart the MQI channel.

  If you ever need to change any configuration information, therefore, you must ensure that all incomplete units of work have been resolved before making the changes. Alternatively, you must ensure that the configuration changes do not affect the ability of the extended transactional client to restart an MQI channel using the information recorded by MTS. The following are examples of such configurations changes:
  - Changing the value of the MQSERVER environment variable
  - Changing entries in the client channel definition table
  - Deleting a server connection channel definition
- Note the following conditions when using an extended transactional client with MTS:

– Within a single thread, a client application can be connected to only one queue manager at a time.
– Each thread of a client application can connect to a different queue manager.
– A client application cannot use shared connection handles.

WebSphere Application Server

If you are using WebSphere MQ JMS, with WebSphere Application Server as your transaction manager, you must perform the following configuration tasks:

- Configure WebSphere MQ Java for use in client mode as described in WebSphere MQ Using Java.
- Configure the interface between WebSphere MQ JMS and WebSphere Application Server. WebSphere MQ Using Java describes how to do this for WebSphere MQ JMS applications that connect to a queue manager in bindings mode. For client mode, you configure the interface in exactly the same way.

To complete the configuration for the extended transactional client, you must then perform the following task. The procedure you follow depends on which version of WebSphere Application Server you are using. In each procedure, you need to enter the fully qualified path name of the Java archive file, com.ibm.mqetclient.jar, which is installed with the extended transactional function. See Table 18 for this information.

Table 18. The location of the com.ibm.mqetclient.jar file

Platform	**Location of com.ibm.mqetclient.jar**
AIX	/usr/mqm/java/lib/com.ibm.mqetclient.jar
HP-UX Linux Solaris	/opt/mqm/java/lib/com.ibm.mqetclient.jar
Windows systems	C:\Program Files\IBM\WebSphere MQ\Java\lib\com.ibm.mqetclient.jar [1]

Note:
1. This is the default location for the file, but you might have installed it in a different location.

For WebSphere Application Server Version 4:

1. Open the WebSphere Application Server Administrative Console.
2. In the navigation pane on the left side of the window, expand the following items in sequence:
   a. **WebSphere Administrative Domain**
   b. **Nodes**
   c. The name of the system that contains the application server in which you want to run your WebSphere MQ JMS applications
   d. **Application Servers**

   Then click the application server in which you want to run your WebSphere MQ JMS applications (**Default Server**, for example).
3. In the right side of the window, click the **JVM Settings** tab.
4. In the System Properties section of the page, click **Add**. An empty row is created in the System Properties table.

5. In the **Name** column of the empty row, type `ws.ext.dirs`.
6. In the **Value** column of the empty row, type the fully qualified path name of the Java archive file, com.ibm.mqetclient.jar.
7. Click **Apply**.
8. Restart the application server for the changes to take effect.

For WebSphere Application Server Version 5:

1. Start the WebSphere Application Server Administrative Console in your Web browser and log in.
2. In the navigation frame on the left side of the page, expand **Servers**.
3. Click the **Application Servers** link. A list of application servers is displayed on the right side of the page.
4. Click the link for the application server in which you want to run your WebSphere MQ JMS applications (**server1**, for example). A page containing the properties of the application server is displayed. Ensure the configuration properties are displayed by clicking the **Configuration** tab, if necessary.
5. In the Additional Properties section of the page, click the **Process Definition** link. The Process Definition page is displayed.
6. In the Additional Properties section of the page, click the **Java Virtual Machine** link. The Java Virtual Machine page is displayed.
7. In the Additional Properties section of the page, click the **Custom Properties** link. The Custom Properties page is displayed.
8. Click **New**. A page to enter the name and value of a custom property is displayed.
9. In the **Name** field, type `ws.ext.dirs`.
10. In the **Value** field, type the fully qualified path name of the Java archive file, com.ibm.mqetclient.jar.
11. Click **OK**.
12. Click **Save** to save the changes to the configuration. A page containing a confirmation prompt is displayed.
13. Click **Save** to confirm the changes.
14. Restart the application server for the changes to take effect.

Verifying the installation

You can verify your WebSphere MQ client and server installation using the supplied sample PUT and GET programs.

These verify that your installation has been completed successfully and that the communication link is working.

This chapter tells you how to use the supplied sample PUT and GET programs to verify that a WebSphere MQ client has been installed correctly, by guiding you through the following tasks:

1. "Setting up the server" on page 76
2. "Setting up the WebSphere MQ client" on page 79
3. "Putting a message on the queue" on page 80
4. "Getting the message from the queue" on page 81
5. "Ending verification" on page 81

The installation used for the example

These instructions assume that:

- The full WebSphere MQ server product has been installed on a server, including the Client Attachment feature on z/OS.
- The WebSphere MQ client software has been installed on a client machine, and WebSphere MQ client files have been transferred, where necessary.

The transmission protocol used in the example is TCP/IP. It is assumed that you have TCP/IP configured on the server and the WebSphere MQ client machines. There is more information about this in "Configuring communication links" on page 61.

Compiled samples AMQSPUTC and AMQSGETC are included in the WebSphere MQ client directories that you installed.

What the example shows

The following example shows how to create a queue manager called *queue.manager.1* (not on z/OS), a local queue called *QUEUE1*, and a server-connection channel called *CHANNEL1* on the server.

It shows how to create the client-connection channel on the WebSphere MQ client workstation; and how to use the sample programs to put a message onto a queue, and then get the message from the queue.

Note: WebSphere MQ object definitions are case-sensitive. You must type the examples ***exactly*** as shown.

Security:

The example does ***not*** address any client security issues. See "Setting up WebSphere MQ client security" on page 83 for details if you are concerned with WebSphere MQ client security issues.

Setting up the server

Setting up the server on UNIX and Windows systems

This section does not apply to i5/OS or z/OS. For information about setting up the server on these platforms, see the sections that follow.

Note: You can use the WebSphere MQ Explorer as well as the command line to carry out most WebSphere MQ tasks. Details of how to use WebSphere MQ Explorer for the verification example are given in "Setting up a Windows server using WebSphere MQ Explorer."

Setting up a Windows server using WebSphere MQ Explorer:

- Create a queue manager
  1. Open WebSphere MQ Explorer from the Start Menu.
  2. Right-click the folder called Queue Managers, select New, and select Queue Manager.
  3. In the first entry field, type the queue manager name, *queue.manager.1*, and click Finish.
- Create a local queue

1. Expand the queue manager you have just created and right-click queues.
2. Select New and select Local Queue.
3. Enter the queue name, QUEUE1, and click Finish.

- Define the server-connection channel
  1. Expand Advanced, and right-click Channels.
  2. Select New and select Server Connection Channel.
  3. Enter the channel name, CHANNEL1, and click Next.
  4. In the dialog navigation pane, click MCA to open the MCA page.
  5. In the MCA User ID field, enter a userid that is a member of the mqm group, typically your own.
  6. Click Finish.
- Run the listener

  The listener will have automatically started when the queue manager was set up. You can verify this by opening Listeners. The listener LISTENER.TCP will be running.

Setting up the server on Unix:

Before you begin

About this task

These instructions assume that no queue manager or other WebSphere MQ objects have been defined. Follow these steps:

Note: WebSphere MQ object definitions are case-sensitive. Any text entered as an MQSC command in lowercase is converted automatically to uppercase unless you enclose it in single quotation marks. Make sure that you type the examples exactly as shown.

1. Create a default queue manager called `saturn.queue.manager` by entering the following command:
   ```
   crtmqm -q queue.manager.1
   ```
   You will see messages telling you that the queue manager has been created, and that the default WebSphere MQ objects have been created.
2. To start the queue manager, type:
   ```
   strmqm
   ```
   A message tells you when the queue manager has started.
3. Enable MQSC commands by typing:
   ```
   runmqsc
   ```
   A message tells you that an MQSC session has started. MQSC has no command prompt.
4. Define a local queue called `QUEUE1` by entering the following command:
   ```
   define qlocal (queue1)
   ```
   A message tells you when the queue has been created.
5. Define a server-connection channel by entering the following command on one line:
   ```
   define channel (channel1) chltype (svrconn) trptype (tcp) mcauser ('mqm')
   ```

A message tells you when the channel has been created.

6. Define a listener by entering the following command:

   Note: If you do not specify the port that the listener should listen on, by omitting the port parameter from the command below, the default of 1414 is used. If you want to specify a port other than 1414, you must include the port parameter in the command, as shown.

   ```
   define listener (listener1) trptype (tcp) control (qmgr) port (port_number)
   ```

   Where

   port_number
   : is the name of the port the listener should run on. This must be the same as the number used when defining your client-connection channel in Setting up the client workstation.

7. Start the listener by entering the following command:

   ```
   start listener (listener1)
   ```

8. Stop MQSC by typing:

   ```
   end
   ```

   You will see some messages, followed by the command prompt.

Results

You have now defined the following objects on the server:

- A default queue manager called `queue.manager.1`
- A local queue called `QUEUE1`
- A listener called `LISTENER1`
- A server-connection channel called `CHANNEL1`

Setting up the server on i5/OS

These instructions assume that no queue manager or other WebSphere MQ objects have been defined. Follow these steps:

1. Create a queue manager by entering the following command:

   ```
   CRTMQM MQMNAME('queue.manager.1')
   ```

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

   ```
   STRMQM MQMNAME('queue.manager.1')
   ```

3. Create a local queue by entering the following command:

   ```
   CRTMQMQ QNAME(QUEUE1)  QTYPE(*LCL)
   ```

4. Create a server-connection channel by entering the following command:

   ```
   CRTMQMCHL CHLNAME(CHANNEL1) CHLTYPE(*SVRCN) TRPTYPE(*TCP)
             MCAUSRID('QMQM')
   ```

   Note: `QMQM` is the default user ID.

5. Start the listener by entering the following command:

   ```
   STRMQMLSR MQMNAME('queue.manager.1')
   ```

Setting up the server on z/OS

Customize your WebSphere MQ for z/OS installation as described in the WebSphere MQ for z/OS System Setup Guide. This includes defining the default system objects and enabling distributed queuing. You do not require the

Batch/TSO, CICS, or IMS™ adapters to run as servers for WebSphere MQ applications running on a client. However, depending on how you choose to issue commands, you might need the Batch/TSO adapter and the operations and control panels to perform administration for clients.

Follow the steps below. You can use any of the valid command input methods to issue the WebSphere MQ commands (MQSC) shown.

1. Start the queue manager by entering the following command:
   ```
   START QMGR
   ```
2. Create a local queue by entering the following command:
   ```
   DEFINE QLOCAL(QUEUE1)
   ```
3. Create a server-connection channel by entering the following command:
   ```
   DEFINE CHANNEL(CHANNEL1) CHLTYPE(SVRCONN) TRPTYPE(TCP) MCAUSER(' ')
   ```
4. Start the channel initiator by entering the following command:
   ```
   START CHINIT
   ```
5. Start the listener by entering the following command:
   ```
   START LSTR TRPTYPE(TCP) PORT(port)
   ```

Setting up the WebSphere MQ client

When a WebSphere MQ application is run on the WebSphere MQ client, it requires the name of the MQI channel, the communication type, and the address of the server to be used. You provide this by defining a client-connection channel. The name used must be same as the name used for the server-connection channel defined on the server. In this example, two methods are used to define the client-connection channel;

- "Defining a client-connection channel using MQSERVER"
- "Defining a client-connection channel using WebSphere MQ Explorer" on page 80

Before starting, type `ping server-address` (where `server-address` is the TCP/IP host name of the server) to confirm that your WebSphere MQ client and server TCP/IP sessions are correctly configured. You can use the network address, in IPv4 dotted decimal form (for example 9.20.9.30) or IPv6 hexadecimal form (for example fe80:43e4:0204:acff:fe97:2c34:fde0:3485), in the `ping` command instead of the host name.

If the `ping` command fails, check that your TCP/IP software is correctly configured.

Defining a client-connection channel using MQSERVER

Create a client-connection channel by setting the MQSERVER environment variable. (For more information, see "Using WebSphere MQ environment variables" on page 111).

- For Windows clients, enter the following command:
  ```
  SET MQSERVER=CHANNEL1/TCP/server-address(port)
  ```
- For UNIX clients, enter the following command:
  ```
  export MQSERVER=CHANNEL1/TCP/'server-address(port)'
  ```

Where:

- `server-address` is the TCP/IP host name of the server

- `(port)` is the TCP/IP port number the server is listening on

If you do not give a port number, WebSphere MQ uses the one specified in the qm.ini file, or the client configuration file. If no value is specified in the qm.ini file, or the client configuration file, WebSphere MQ uses the port number identified in the TCP/IP services file for the service name `MQSeries`. If this entry in the services file does not exist, a default value of 1414 is used. It is important that the port number used by the client and the port number used by the server listener program are the same.

Defining a client-connection channel using WebSphere MQ Explorer

You can use WebSphere MQ Explorer to define the client-connection if you are setting up the client and server on the same machine. The following steps will define the client-connection end of the channel.

1. Select the queue manager, *queue.manager.1*
2. Select **Advanced**, then **Client Connection**, then **New**, then **Client Connection Channel**.

   Enter the channel name, CHANNEL1, for the client connection, and click Next
3. Enter the queue manager name, *queue.manager.1*
4.

   Enter the following as the connection name:

   ```
   server-address(port)
   ```

   Where:
   - `server-address` is the TCP/IP host name of the server
   - `(port)` is the TCP/IP port number the server is listening on
5. Click Finish.
6. Set the MQCHLLIB environment variable:
   - For Windows clients, enter the following command:

     ```
     SET MQCHLLIB=mqmtop\qmgrs\queue.manager.1\@ipcc
     ```
   - For UNIX clients, enter the following command:

     ```
     export MQCHLLIB=mqmtop/qmgrs/queue.manager.1/@ipcc
     ```

Putting a message on the queue

On the WebSphere MQ client workstation, put a message on the queue using the amqsputc sample program.

Information on errors at this stage can be found in “Error messages with WebSphere MQ clients” on page 144.

On the WebSphere MQ client workstation

1. From a command prompt window, change to the directory containing the sample program amqsputc.exe. This is in the `mqmtop/samp/bin` directory on UNIX and the `mqmtop\bin` directory in Windows (where mqmtop is the MQ install directory). Then enter the following command:

   ```
   amqsputc QUEUE1 queue.manager.1
   ```

   where `queue.manager.1` is the name of the queue manager on the server and `QUEUE1` is the queue you set up on the server in an earlier step.
2. The following message is displayed:

```
Sample AMQSPUT0 start
target queue is QUEUE1
```

3. Type some message text and then press Enter twice.
4. The following message is displayed:

   ```
   Sample AMQSPUT0 end
   ```

5. The message is now on the queue.

Getting the message from the queue

On the WebSphere MQ client workstation, get a message from the queue using the amqsgetc sample program.

On the WebSphere MQ client workstation

1. Change to the directory containing the sample programs, and then enter the following command:

   ```
   amqsgetc QUEUE1 QM1
   ```

   Where `QM1` is the name of the queue manager on the server.
2. The message on the queue is removed from the queue and displayed.

Ending verification

The verification process is now complete.

On z/OS, you can stop the queue manager on the server by issuing the `STOP CHINIT` command followed by the `STOP QMGR` command.

On other platforms, you can stop the queue manager on the server by typing:

```
endmqm queue.manager.1
```

If you want to delete the queue manager on the server (not z/OS) type:

```
dltmqm queue.manager.1
```

Chapter 3. System administration

Setting up WebSphere MQ client security

You must consider WebSphere MQ client security, so that the client applications do not have unrestricted access to resources on the server.

There are two aspects to security between a client application and its queue manager server: authentication and access control.

Note: WebSphere MQ provides Secure Sockets Layer (SSL) support for WebSphere MQ clients on the following platforms:
- UNIX systems
- Windows

For more information on SSL support for WebSphere MQ queue managers and WebSphere MQ clients, see WebSphere MQ Security.

Authentication

There are three levels of security to consider, as shown in the following diagram. MCA is a Message Channel Agent.

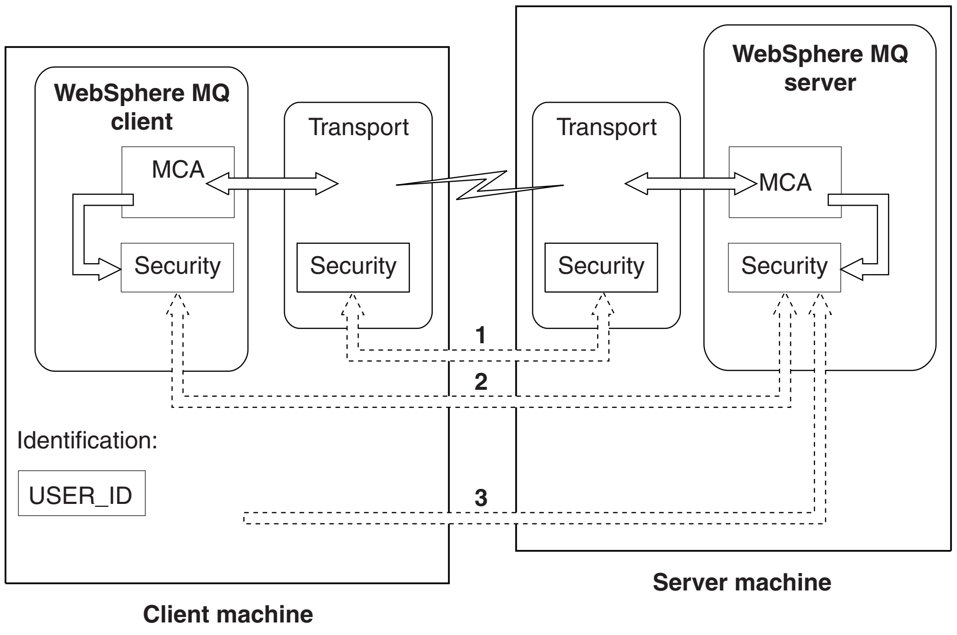

Figure 20. Security in a client-server connection

1. Transport level

   This is the same as for two WebSphere MQ queue managers (server to server) and is described in the WebSphere MQ Intercommunication manual.
2. Channel security exits

   The channel security exits for client to server communication can work in the same way as for server to server communication. A protocol independent pair of exits can be written to provide mutual authentication of both the client and the server. A full description is given in the WebSphere MQ Intercommunication manual.

3. Identification passed to a channel security exit

   In client to server communication, the channel security exits do not have to operate as a pair. The exit on the WebSphere MQ client side can be omitted. In this case the user ID is placed in the channel descriptor (MQCD) and the server-side security exit can alter it, if required. Windows clients also send additional information to assist identification.

   - The user ID that is passed to the server is the currently logged-on user ID on the client. In addition, a client on Windows passes the security ID of the currently logged-on user.

   The values of the user ID and, if available, the security ID, can be used by the server security exit to establish the identity of the WebSphere MQ client.

User IDs

If the WebSphere MQ client is on Windows and the WebSphere MQ server is also on Windows and has access to the domain on which the client user ID is defined, WebSphere MQ supports user IDs of up to 20 characters.

A WebSphere MQ for Windows server does not support the connection of a Windows client if the client is running under a user ID that contains the @ character, for example, `abc@d`. The return code to the **MQCONN** call at the client is MQRC_NOT_AUTHORIZED.

On all other platforms and configurations, the maximum length for user IDs is 12 characters.

Access control

Access control in WebSphere MQ is based upon user IDs. The user ID of the process making MQI calls is normally used. In the case of MQ Clients, the server-connection MCA makes MQI calls on behalf of MQ Clients. You can select an alternative user ID for the server-connection MCA to use for making MQI calls. The alternative user ID can be associated either with the client workstation, or with anything you choose to organize and control the access of clients. The user ID needs to have the necessary authorities allocated to it on the server to issue MQI calls. Choosing an alternative user ID is preferable to allowing clients to make MQI calls with the authority of the server-connection MCA.

As the server-connection MCA makes MQI calls on behalf of remote users, it is important to consider the security implications of the server-connection MCA issuing MQI calls on behalf of remote clients and how to administer the access of a potentially very large number of users.

- One approach is for the server-connection MCA to issue MQI calls on its own authority. But beware, it is normally undesirable for the server-connection MCA, with its powerful access capabilities, to issue MQI calls on behalf of client users.
- Another approach is to make use of the user ID that flows from the client. The server-connection MCA can issue MQI calls using the access capabilities of the client user ID. This presents a number of questions to consider:
  1. There are different formats for the user ID on different platforms. This sometimes gives rise to problems if the format of the user ID on the client differs from the acceptable formats on the server.

2. There are potentially very many clients, with different and changing user IDs. The IDs need to be defined and managed on the server.
3. Is the user ID to be trusted? Any user ID can be flowed from a client, not necessarily the ID of the logged on user. For example the client might flow an ID with full mqm authority that was intentionally *only* defined on the server for security reasons.

- The preferred approach is to define client identification tokens at the server, and so limit the capabilities of client connected applications. This is typically done by setting the server-connection channel property MCAUSER to a special user ID value to be used by clients, and defining a small number of IDs for use by clients with different level of authorization on the server.

For WebSphere MQ clients, the process that issues the MQI calls is the server-connection MCA. The user ID used by the server-connection MCA is contained in either the `MCAUserIdentifier` or `LongMCAUserIdentifier` fields of the MQCD. The contents of these fields are set by:

- Any values set by security exits
- The user ID from the client
- MCAUSER (in the server-connection channel definition)

The security exit can override the values that are visible to it, when it is invoked.

- If the server-connection channel MCAUSER attribute is set to nonblank, the MCAUSER value is used.
- If the server-connection channel MCAUSER attribute is blank, the user ID received from the client is used.
- If the server-connection channel MCAUSER attribute is blank, and no user ID is received from the client then the user ID that started the server-connection channel is used.

When the user ID fields are derived from the user ID that started the server-connection channel, the following value is used:

- For z/OS, the user ID assigned to the channel initiator started task by the z/OS started procedures table. See the WebSphere MQ for z/OS System Setup Guide for more information.
- For TCP/IP (non-z/OS), the user ID from the inetd.conf entry, or the user ID that started the listener.
- For SNA (non-z/OS), the user ID from the SNA Server entry or (if there is none) the incoming attach request, or the user ID that started the listener.
- For NetBIOS or SPX, the user ID that started the listener.

If any server-connection channel definitions exist that have the MCAUSER attribute set to blank, clients can use this channel definition to connect to the queue manager with access authority determined by the user ID supplied by the client. This might be a security exposure if the system on which the queue manager is running allows unauthorized network connections. The WebSphere MQ default server-connection channel (SYSTEM.DEF.SVRCONN) has the MCAUSER attribute set to blank. **To prevent unauthorized access**, update the MCAUSER attribute of the default definition with a user ID that has no access to WebSphere MQ objects.

When you define a channel with `runmqsc`, the MCAUSER attribute is changed to uppercase unless the user ID is contained within single quotation marks.

For servers on UNIX systems and Windows, the content of the `MCAUserIdentifier` field that is received from the client is changed to lowercase.

For servers on i5/OS, the content of the `LongMCAUserIdentifier` field that is received from the client is changed to uppercase.

For servers on UNIX systems, the content of the `LongMCAUserIdentifier` field that is received from the client is changed to lowercase.

Using channels

This chapter discusses the following topics:
- "What is a channel?"
- "Defining MQI channels" on page 89
- "Creating a queue manager and starting MQSC on the server" on page 90
- "Creating one definition on the WebSphere MQ client and the other on the server" on page 90
- "Creating both definitions on the server" on page 93
- "Channel exits" on page 96
- "Connecting a client to a queue-sharing group" on page 98
- "Migration and client channel definition tables" on page 95

What is a channel?

A channel is a logical communication link between a WebSphere MQ client and a WebSphere MQ server, or between two WebSphere MQ servers. A channel has two definitions: one at each end of the connection. The same *channel name* must be used at each end of the connection, and the *channel type* used must be compatible.

There are two categories of channel in WebSphere MQ, with different channel types within these categories:

Message Channels

A message channel is a one-way link. It connects two queue managers via *message channel agents* (MCAs). Its purpose is to transfer messages from one queue manager to another. Message channels are not required by the client server environment. There is more information on message channels in WebSphere MQ Intercommunication.

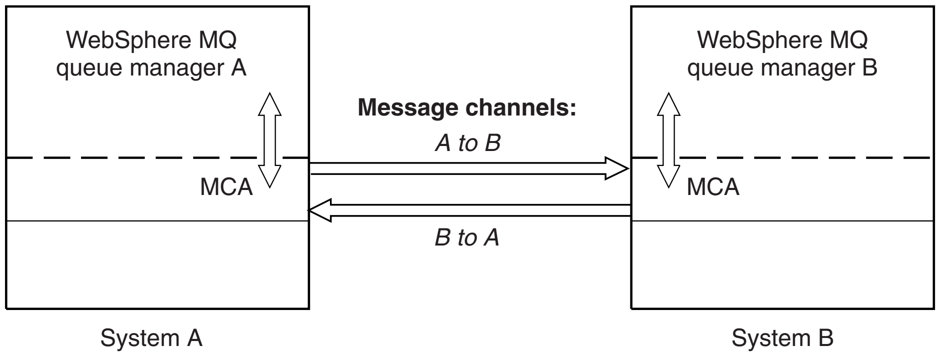

Figure 21. Message channels between two queue managers

MQI Channels

An MQI channel connects a WebSphere MQ client to a queue manager on a server machine, and is established when you issue an **MQCONN** or **MQCONNX** call from a WebSphere MQ client application. It is a two-way link and is used for the transfer of MQI calls and responses only, including **MQPUT** calls that contain message data and **MQGET** calls that result in the return of message data. There are different ways of creating and using the channel definitions (see “Defining MQI channels” on page 89).

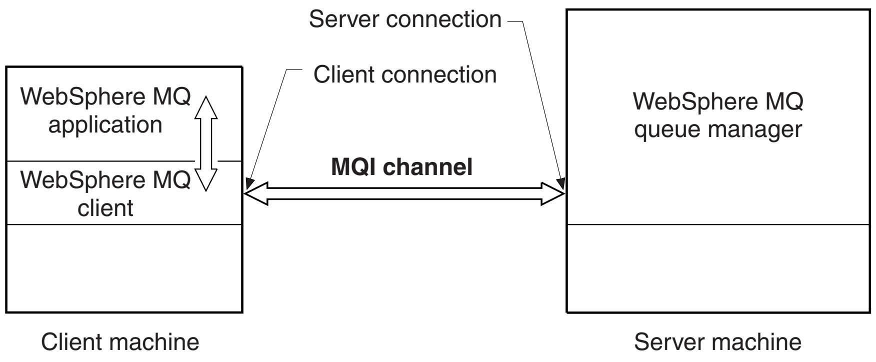

Figure 22. Client-connection and server-connection on an MQI channel

An MQI channel can be used to connect a client to a single queue manager, or to a queue manager that is part of a queue-sharing group (see “Connecting a client to a queue-sharing group” on page 98).

There are two channel types for MQI channel definitions. They define the bi-directional MQI channel.

Client-connection channel
This type is for the WebSphere MQ client.

Server-connection channel
| This type is for the server running the queue manager, with which the
| WebSphere MQ application, running in a WebSphere MQ client
| environment, will communicate.

Migrating client-connection and server-connection channels

This topic describes issues relating to migrating client-connection or server-connection channels to WebSphere MQ Version 7.0.

The default for client channels is to use sharing conversations. On client channels only using sharing conversations heartbeats can flow across the channel (from both ends) all the time, not just when an MQGET is outstanding. This might affect whether you set heartbeating on client channels and what value you set it to.

The default value on a migrated client-connection or server-connection channel for the `SHARECNV` channel attribute is 10, and the default WebSphere MQ V7.0 sharing MQCNO value for an existing client application is `MQCNO_ALL_CONVS_SHARE`. Conversations are, therefore, shared on TCP/IP channel instances by default when you migrate to WebSphere MQ Version 7.0. This is not apparent, unless you have send, receive, or security exits written to perform actions that are not generally expected in send, receive or security exits. In particular:

- If send or receive exits alter the MQCD structure on an `MQXR_INIT` call, the effect of these exits will differ, depending on whether they are on the first, or subsequent conversations on a channel instance with sharing conversations:
  - If the MQCXP `SharingConversations` field is set to FALSE, this exit instance applies to the first conversation on the channel instance. No other exit can be changing the MQCD at the same time and changes that are made to the MQCD can affect the way that the channel runs.
  - If the MQCXP `SharingConversations` field is set to TRUE, this exit instance applies to a conversation that is sharing the channel instance with other conversations. Changes made to the MQCD in the exit instance are retained in the MQCD but cannot affect the way the channel runs.
- If send, receive, or security exit instances alter the MQCD, when the MQCXP `SharingConversations` field is set to TRUE, exit instances on other conversations can be changing the MQCD at the same time; it might be necessary to serialize access to the MQCD across these different exit instances.
- For further information on the MQCXP `SharingConversations` field, see Sharing Conversations*WebSphere MQ Intercommunication*.
- If send or receive exits on the server-connection side of the channel instance need to perform MQI calls on the connection with which they are associated, they use the connection handle provided in the MQCXP `Hconn` field. You must be aware that client-connection send and receive exits cannot make MQI calls.

Performance implications of sharing conversations on client-connection channels

The SHARECNV channel attribute specifies the maximum number of conversations that share each TCP/IP channel instance. Sharing conversations has performance implications when migrating client-connection channels to WebSphere MQ Version 7.0.

A SHARECNV channel attribute of 0 specifies no sharing of conversations over a TCP/IP channel instance. The channel instance runs in a mode similar to that of WebSphere MQ Version 6.0 and inhibits the following WebSphere MQ Version 7.0 features:

- Administrator stop-quiesce
- Heartbeating
- Read ahead

- Client asynchronous consume

Established client applications that do not need to use the WebSphere MQ Version 7.0 features can use SHARECNV(0). Using SHARECNV(0), processing of messages is, on average, 3 percent slower than under WebSphere MQ Version 6.0.

A SHARECNV channel attribute value of 1 specifies no sharing of conversations over a TCP/IP channel instance but makes the WebSphere MQ Version 7.0 features available.

A SHARECNV channel attribute value greater than 1 specifies sharing of conversations over a TCP/IP channel instance and makes the WebSphere MQ Version 7.0 features available.

The default value on a migrated client-connection channel for the SHARECNV channel attribute is 10, and the default WebSphere MQ V7.0 sharing MQCNO value for an existing client application is MQCNO_ALL_CONVS_SHARE. Conversations are, therefore, shared on TCP/IP channel instances by default when you migrate to WebSphere MQ Version 7.0.

All of the conversations on a socket are received by the same thread. High SHARECNV limits have the advantage of reducing queue manager thread usage. However, if a large number of conversations sharing a socket are all busy, there is a possibility of delays as the conversations contend with one another to use the receiving thread. In this situation a lower SHARECNV value is better.

On average, processing of messages from client applications is 15 percent slower when using SHARECNV(10), compared with SHARECNV(0); this is the cost of providing the client function that is new in WebSphere MQ Version 7.0. However, performance improvements introduced in WebSphere MQ Version 7.0 on distributed platforms and the availability of WebSphere MQ Version 7.0 performance features such as read ahead and client asynchronous consume, can mitigate this performance difference.

You must bear in mind the preceding points when optimizing the SHARECNV value for your enterprise.

Defining MQI channels

To create a new channel, you have to create **two** channel definitions, one for each end of the connection, using the same channel name and compatible channel types. In this case, the channel types are *server-connection* and *client-connection*.

Automatically defined channels

WebSphere MQ products on platforms other than z/OS include a feature that can automatically create a channel definition on the server if one does not exist.

If an inbound attach request is received from a client and an appropriate server-connection definition cannot be found on that queue manager, WebSphere MQ creates a definition automatically and adds it to the queue manager. The automatic definition is based on the definition of the default server-connection channel SYSTEM.AUTO.SVRCONN. You enable automatic definition of server-connection definitions by updating the queue manager object using the ALTER QMGR command with the CHAD parameter (or the PCF command Change Queue Manager with the ChannelAutoDef parameter).

For more information about the automatic creation of channel definitions, see the WebSphere MQ Intercommunication book.

User defined channels

When the server does not automatically define channels there are two ways of creating the channel definitions and giving the WebSphere MQ application on the WebSphere MQ client machine access to the channel.

These two methods are described in detail in this chapter:

1. Create one channel definition on the WebSphere MQ client and the other on the server.

   This applies to any combination of WebSphere MQ client and server platforms. Use it when you are getting started on the system, or to test your setup.

   See "Creating one definition on the WebSphere MQ client and the other on the server" for details on how to use this method.
2. Create both channel definitions on the server machine.

   Use this method when you are setting up multiple channels and WebSphere MQ client machines at the same time.

   See "Creating both definitions on the server" on page 93 for details on how to use this method.

Whichever method you choose, first you will need to create a queue manager and start WebSphere MQ Script (MQSC) commands (see "Creating a queue manager and starting MQSC on the server").

Creating one definition on the WebSphere MQ client and the other on the server

On all platforms, you can use WebSphere MQ Script (MQSC) commands, programmable command format (PCF) commands, or the WebSphere MQ Explorer to define a server-connection channel on the server machine.

On z/OS you can also use the Operation and Control panels and on i5/OS you can also use the panel interface. Because MQSC commands are not available on a machine where WebSphere MQ has been installed as a WebSphere MQ client only, you must use different ways of defining a client-connection channel on the client machine.

On the server

If your server platform is not z/OS, you first create and start a queue manager and then start MQSC commands.

Creating a queue manager and starting MQSC on the server:

1. Create a queue manager, called QM1 for example:
   ```
   crtmqm QM1
   ```
2. Start the queue manager:
   ```
   strmqm QM1
   ```
3. Start MQSC commands.
   ```
   runmqsc QM1
   ```

Defining the server-connection channel:

Define a channel with your chosen name and a channel type of *server-connection*. This channel definition is associated with the queue manager running on the server.

For example:

```
DEFINE CHANNEL(CHAN1) CHLTYPE(SVRCONN) TRPTYPE(TCP) +
DESCR('Server-connection to Client_1')
```

On the WebSphere MQ client

There are two ways of defining a client-connection channel on the client machine.

Using MQSERVER:

You can use the MQSERVER environment variable to specify a simple definition of a client-connection channel. It is simple in the sense that you can specify only a few attributes of the channel using this method.

- Specify a simple channel definition on Windows as follows:
  ```
  SET MQSERVER=ChannelName/TransportType/ConnectionName
  ```
- Specify a simple channel definition on UNIX systems as follows:
  ```
  export MQSERVER=ChannelName/TransportType/ConnectionName
  ```

Where:

- `ChannelName` must be the same name as defined on the server. It cannot contain a forward slash.
- `TransportType` can be one of the following, depending on your WebSphere MQ client platform:
  - LU62
  - TCP
  - NETBIOS
  - SPX

  Note: On UNIX systems the TransportType is case sensitive and must be uppercase. An **MQCONN** or **MQCONNX** call will return 2058 if the TransportType is not recognized
- `ConnectionName` is the name of the server machine as defined to the communications protocol (TransportType).

For example, on Windows:

```
SET MQSERVER=CHANNEL1/TCP/MCID66499
```

or, on a UNIX system:

```
export MQSERVER=CHANNEL1/TCP/'MCID66499'
```

Note: To change the TCP/IP port number, see “MQSERVER” on page 114.

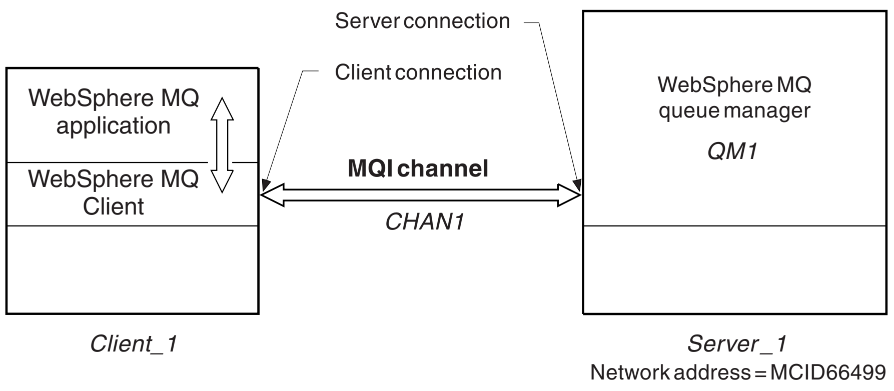

Figure 23. Simple channel definition

Some more examples of a simple channel definition on Windows are:

```
SET MQSERVER=CHANNEL1/TCP/9.20.4.56
SET MQSERVER=CHANNEL1/NETBIOS/BOX643
```

Some examples of a simple channel definition on a UNIX system are:

```
export MQSERVER=CHANNEL1/TCP/'9.20.4.56'
export MQSERVER=CHANNEL1/LU62/BOX99
```

Where `BOX99` is the LU 6.2 ConnectionName.

On the WebSphere MQ client, all **MQCONN** or **MQCONNX** requests then attempt to use the channel you have defined, unless the channel is overridden in an MQCD structure referenced from the MQCNO structure supplied to **MQCONNX**.

Note: For more information on the MQSERVER environment variable see "Using WebSphere MQ environment variables" on page 111.

Using the MQCNO structure on an MQCONNX call:

A WebSphere MQ client application can use the connect options structure, MQCNO, on an MQCONNX call to reference a channel definition structure, MQCD, that contains the definition of a client-connection channel.

In this way, the client application can specify the *ChannelName*, *TransportType*, and *ConnectionName* attributes of a channel at run time, and this enables the client application to connect to multiple server queue managers simultaneously. This is not possible if you define a channel using the MQSERVER environment variable.

A client application can also specify attributes of a channel such as *MaxMsgLength* and *SecurityExit*. This allows the client application to specify values for the attributes that are not the default values, and allows channel exit programs to be called at the client end of an MQI channel.

If a channel uses the Secure Sockets Layer (SSL), a client application can also provide information relating to SSL in the MQCD structure. Additional information relating to SSL can be provided in the SSL configuration options structure, MQSCO, which is also referenced by the MQCNO structure on an MQCONNX call.

For more information about the MQCNO, MQCD, and MQSCO structures, see the WebSphere MQ Application Programming Reference.

Note: A sample connect program called amqscnxc demonstrates the use of this function.

Creating both definitions on the server

First define a server-connection channel and then define a client-connection channel.

On all platforms, you can use WebSphere MQ Script (MQSC) commands, programmable command format (PCF) commands or the WebSphere MQ Explorer to define a server-connection channel on the server machine. On z/OS you can also use the Operation and Control panels and on i5/OS you can also use the panel interface.

Defining the server-connection channel

1. If your server platform is not z/OS, you first create and start a queue manager and then start MQSC commands. See "Creating a queue manager and starting MQSC on the server" on page 90
2. On the server machine, define a channel with your chosen name and a channel type of *server-connection*.

   For example:

   ```
   DEFINE CHANNEL(CHANNEL2) CHLTYPE(SVRCONN) TRPTYPE(TCP) +
   DESCR('Server-connection to Client_2')
   ```

   This channel definition is associated with the queue manager running on the server.

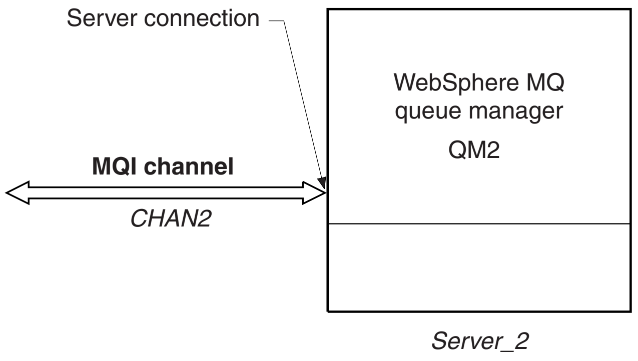

Figure 24. Defining the server-connection channel

Defining the client-connection channel

Define a channel with the ***same*** name and a channel type of *client-connection*.

You must state the connection name (CONNAME). For TCP/IP this is the network address or host name of the server machine. It is also advisable to specify the queue manager name (QMNAME) to which you want your WebSphere MQ application, running in the client environment, to connect. See "Running applications on WebSphere MQ clients" on page 129.

For example:

```
DEFINE CHANNEL(CHANNEL2) CHLTYPE(CLNTCONN) TRPTYPE(TCP) +
CONNAME(9.20.4.26) QMNAME(QM2) DESCR('Client-connection to Server_2')
```

On platforms other than z/OS, this channel definition is generally stored in a file called the client channel definition table, which is associated with the queue manager running on the server. The client channel definition table can contain more than one client-connection channel definition. For more information about the client channel definition table, and for the corresponding information about how client-connection channel definitions are stored on z/OS, see "Client channel definition table."

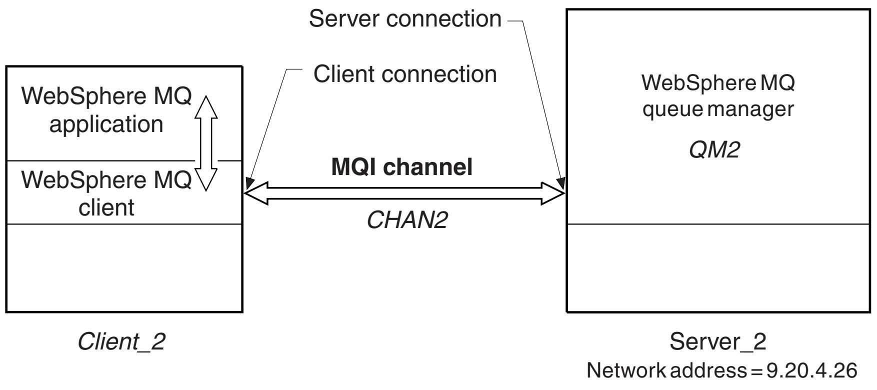

Figure 25. Defining the client-connection channel

Accessing client-connection channel definitions

If the client channel definition table on the server machine cannot be accessed from the client machine as a shared file, you must copy the client channel definition table to the client machine as a binary file.

| On the client machine, you can then use the client configuration file, or the environment variables, MQCHLLIB and MQCHLTAB, to specify the location and name of the file containing the client channel definition table. The WebSphere MQ client uses the values of these environment variables to access the client channel definition table. See "MQCHLLIB" on page 112 and "MQCHLTAB" on page 113 for more information.

For example, you can set the environment variables on a UNIX system by typing:

```
export MQCHLLIB=/mqmtop/qmgrs/QUEUEMANAGERNAME/@ipcc
export MQCHLTAB=AMQCLCHL.TAB
```

As an alternative to using environment variables MQCHLLIB and MQCHLTAB on Windows systems, you can use the **setmqscp** control command to publish the client-connection channel definitions in Active Directory. For information about this command and its syntax, see the WebSphere MQ System Administration Guide.

Note: If the MQSERVER environment variable is set, a WebSphere MQ client uses the client-connection channel definition specified by MQSERVER in preference to any definitions in the client channel definition table.

Client channel definition table

How to locate the file containing the client channel definition table.

On platforms other than z/OS, the client-connection channel definition described previously is stored in the *client channel definition table* associated with the queue manager running on the server. The file containing the table is called AMQCLCHL.TAB and is a binary file that cannot be edited directly. You can use the DEFINE CHANNEL command to add a client connection channel to the table, and the ALTER CHANNEL command to alter the attributes of a channel that already has an entry in the table.

Do not delete AMQCLCHL.TAB. It contains default channel definitions that are required when you define a channel. If you suspect that this has been deleted, for example you get error messages when you try to define a new channel, check to see that the file exists.

If you install WebSphere MQ in the default location, AMQCLCHL.TAB is located in the following directory on a server machine:

- On i5/OS, in the Integrated File System (IFS):

  `/QIBM/UserData/mqm/qmgrs/QUEUEMANAGERNAME/&ipcc`
- On UNIX systems:

  `/mqmtop/qmgrs/QUEUEMANAGERNAME/@ipcc`

  Note that the name of the directory referred to by *QUEUEMANAGERNAME* is case sensitive on UNIX systems.
- On Windows, the location of this file defaults to:

  `C:\Program Files\IBM\WebSphere MQ\qmgrs\QUEUEMANAGERNAME\@ipcc`

On z/OS, client-connection channel definitions are stored with WebSphere MQ objects in page set zero. They are not stored in a file that can be accessed from a client machine, nor can they be copied directly to a client machine. Instead, you must use the MAKECLNT parameter of the COMMAND function of the WebSphere MQ utility program, CSQUTIL, to generate a file that contains a client channel definition table. You can then download this file to a client machine using a file transfer program. For details, see the WebSphere MQ for z/OS System Administration Guide.

Here is some sample JCL for making a client channel definition file:

```
//CLIENT    EXEC  PGM=CSQUTIL,PARM='QM2'
//STEPLIB   DD   DISP=SHR,DSN=thlqual.SCSQANLE
//          DD   DISP=SHR,DSN=thlqual.SCSQAUTH
//OUTCLNT   DD   DISP=OLD,DSN=MY.CLIENTS
//SYSPRINT  DD SYSOUT=*
//SYSIN     DD *
COMMAND DDNAME(CMDCHL) MAKECLNT(OUTCLNT) CCSID(437)
/*
//CMDCHL    DD *
DISPLAY CHANNEL(*) ALL TYPE(CLNTCONN)
DISPLAY AUTHINFO (*) ALL
/*
```

where `thlqual` is a high level qualifier for the WebSphere MQ library data sets. The data set for the client channel definition file, identified by the DD name `OUTCLNT` in the sample JCL, must have the format:

```
RECFM=U, LRECL=6144, BLKSIZE=6144
```

If you use FTP to copy the file, remember to type `bin` to set binary mode; do not use the default ASCII mode.

Migration and client channel definition tables

In general, the internal format of the client channel definition table might change from one release level of WebSphere MQ to the next. Because of this, a WebSphere

MQ client can use a client channel definition table only when it has been prepared by a server queue manager that is at the same release level as the client, or at an earlier release level.

A Version 7.0 WebSphere MQ client can use a client channel definition table that has been prepared by a Version 6.0 queue manager but a Version 6.0 client cannot use a client channel definition table that has been prepared by a Version 7.0 queue manager.

Channel exits

The channel exits available to the WebSphere MQ client environment on UNIX systems and Windows are:
- Send exit
- Receive exit
- Security exit

These exits are available at both the client and the server end of the channel. Exits are not available to your application if you are using the MQSERVER environment variable. Exits are explained in the WebSphere MQ Intercommunication manual.

The send and receive exits work together. There are several possible ways in which you can use them:
- Splitting and reassembling a message
- Compressing and decompressing data in a message (this functionality is provided as part of WebSphere MQ, but you might want to use a different compression technique)
- Encrypting and decrypting user data (this functionality is provided as part of WebSphere MQ, but you might want to use a different encryption technique)
- Journaling each message sent and received

You can use the security exit to ensure that the WebSphere MQ client and server machines are correctly identified, as well as to control access to each machine.

Path to exits

On UNIX systems, an `mqclient.ini` file is added to your system (in /var/mqm) during installation of the WebSphere MQ client. A default path for location of the channel exits on the client is defined in this file, using the stanza:

```
ClientExitPath:
  ExitsDefaultPath=<defaultprefix>/exits
```

where `<defaultprefix>` is the value defined for your system in the DefaultPrefix stanza of the `mqclient.ini` file.

On Windows, the default path for location of the channel exits is C:\Program Files\IBM\Websphere MQ\exits. This location is set using the

```
HKEY_LOCAL_MACHINE\SOFTWARE\IBM\MQSeries\CurrentVersion\Configuration\
ClientExitPath\ExitsDefaultPath
```

setting in the registry.

When a channel is initialized, after an **MQCONN** or **MQCONNX** call, the `mqs.ini` file or Registry is searched. The ClientExitPath stanza is read and any channel exits that are specified in the channel definition are loaded.

Security exits on a client connection

You can use security exit programs to verify that the partner at the other end of a channel is genuine. Special consideration apply when a security exit is applied to a client connection.

Figure 26 on page 98 illustrates the use of security exits in a client connection, using the WebSphere MQ Object Authority Manager to authenticate a user. Either SecurityParmsPtr or SecurityParmsOffset is set in the MQCNO structure on the client and there are security exits at both ends of the channel. After the normal security message exchange has ended, and the channel is ready to run, the MQCSP structure accessed from the MQCXP SecurityParms field is passed to the security exit on the client. The exit type is set to MQXR_SEC_PARMS. The security exit can elect to do nothing to the user identifier and password, or it can alter either or both of them. The data returned from the exit is then sent to the server-connection end of the channel. The MQCSP structure is rebuilt on the server-connection end of the channel and is passed to the server-connection security exit accessed from the MQCXP SecurityParms field. The security exit receives and processes this data. This processing will typically be to reverse any change made to the userid and password fields in the client exit, which are then used to authorize the queue manager connection. The resulting MQCSP structure is referenced using SecurityParmsPtr in the MQCNO structure on the queue manager system.

If SecurityParmsPtr or SecurityParmsOffset are set in the MQCNO structure and there is a security exit at only one end of the channel, the security exit will receive and process the MQCSP structure. Actions such as encryption are inappropriate for a single user exit, as there is no exit to perform the complementary action.

If SecurityParmsPtr and SecurityParmsOffset are not set in the MQCNO structure and there is a security exit at either or both ends of the channel, the security exit or exits will be called. Either security exit can return its own MQCSP structure, addressed through the SecurityParmsPtr; the security exit is not called again until it is terminated (ExitReason of MQXR_TERM). The exit writer can free the memory used for the MQCSP at that stage.

When a server-connection channel instance is sharing more than one conversation, the pattern of calls to the security exit is restricted on the second and subsequent conversations.

For the first conversation, the pattern is the same as if the channel instance is not sharing conversations. For the second and subsequent conversations, the security exit is never called with MQXR_INIT, MQXR_INIT_SEC, or MQXR_SEC_MSG. It is called with MQXR_SEC_PARMS.

In a channel instance with sharing conversations, MQXR_TERM is called only for the last conversation running.

Each conversation has the opportunity in the MQXR_SEC_PARMS invocation of the exit to alter the MQCD; on the server-connection end of the channel this can be useful to vary, for example, the MCAUserIdentifier or LongMCAUserIdPtr values before the connection is made to the queue manager.

Server-connection exit	Client-connection exit
Invoked with MQXR_INIT Responds with MQXCC_OK	Invoked with MQXR_INIT Responds with MQXCC_OK
Invoked with MQXR_INIT_SEC Responds with MQXCC_OK	
	Invoked with MQXR_INIT_SEC Responds with MQXCC_SEND_SEC_MSG
Invoked with MQXR_SEC_MSG Responds with MQXCC_SEND_SEC_MSG	
	Invoked with MQXR_SEC_MSG Responds with MQXCC_OK
	Invoked with MQXR_SEC_PARMS
Invoked with MQXR_SEC_PARMS	
Data transfer begins	

Figure 26. Client connection-initiated exchange with agreement for client connection using security parameters

Connecting a client to a queue-sharing group

You can connect a client to a queue-sharing group by creating an MQI channel between a client and a queue manager on a server machine that is a member of a queue-sharing group.

A queue-sharing group is formed by a set of queue-managers that can access the same set of shared queues. For more information on shared queues, see the WebSphere MQ for z/OS Concepts and Planning Guide book and the WebSphere MQ Intercommunication book.

A client putting to a shared queue can connect to any member of the queue-sharing group. The benefits of connecting to a queue-sharing group are possible increases in front-end and back-end availability, and increased capacity. You can connect a client to a queue-sharing group in the following ways.

Connecting to a specific queue manager

Connecting directly to a queue manager in a queue-sharing group gives the benefit that you can put messages to a shared target queue, which increases back-end availability.

Connecting to the generic interface

Connecting to the generic interface of a queue-sharing group will open a session with one of the queue managers in the group.

The generic interface can be a WLM/DNS group name or a VTAM® generic resource name, or another common interface to the queue-sharing group. See WebSphere MQ Intercommunication for more details on setting up a generic interface.

Connecting to the generic interface increases front-end availability, because the client queue manager can connect with any queue-manager in the group. You connect to the group using the generic interface when you do not want to connect to a specific queue manager within the queue-sharing group.

Creating channel definitions:

To connect to the generic interface of a queue-sharing group you need to create channel definitions that can be accessed by any queue manager in the group. To do this you need to have the same definitions on each queue manager in the group.

Define the SVRCONN channel as follows:

```
DEFINE CHANNEL(CHANNEL1) CHLTYPE(SVRCONN) TRPTYPE(TCP) +
MCAUSER(' ') QSGDISP(GROUP)
```

Channel definitions on the server are stored in a shared DB2 repository. Each queue manager in the queue-sharing group makes a local copy of the definition, ensuring that you will always connect to the correct server-connection channel when you issue an MQCONN or MQCONNX call.

Define the CLNTCONN channel as follows:

```
DEFINE CHANNEL(CHANNEL1) CHLTYPE(CLNTCONN) TRPTYPE(TCP) +
CONNAME(WLM/DNS groupname) QMNAME(QSG1) +
DESCR('Client-connection to Queue Sharing Group QSG1') QSGDISP(GROUP)
```

Because the generic interface of the queue-sharing group is stored in the CONNAME field in the client-connection channel, you can now connect to any queue manager in the group, and put to shared queues owned by that group.

Stopping channels

In WebSphere MQ, when you issue a STOP CHANNEL command against a server-connection channel, you can choose what method to use to stop the client-connection channel.

This means that a client channel issuing an MQGET wait call can be controlled, and you can decide how and when to stop the channel.

The STOP CHANNEL command can be issued with three modes, indicating how the channel is to be stopped:

Quiesce
Stops the channel after any current messages have been processed.

If sharing conversations is enabled, the WebSphere MQ client becomes aware of the stop request in a timely manner; this time is dependent upon the speed of the network. The client application becomes aware of the stop request as a result of issuing a subsequent call to WebSphere MQ.

Force Stops the channel immediately.

Terminate
Stops the channel immediately. If the channel is running as a process, it can terminate the channel's process, or if the channel is running as a thread, its thread.

This is a multi-stage process. If mode terminate is used, an attempt is made to stop the server-connection channel, first with mode quiesce, then with mode force, and if necessary with mode terminate. The client can receive different return codes during the different stages of termination. If the process or thread is terminated, the client receives a communication error.

The return codes returned to the application vary according to the MQI call issued, and the STOP CHANNEL command issued. The client will receive either an MQRC_CONNECTION_QUIESCING or an MQRC_CONNECTION_BROKEN return code. If a client detects MQRC_CONNECTION_QUIESCING it should try to complete the current transaction and terminate. This is not possible with MQRC_CONNECTION_BROKEN. If the client does not complete the transaction and terminate fast enough it will get CONNECTION_BROKEN after a few seconds. A STOP CHANNEL command with MODE(FORCE) or MODE(TERMINATE) is more likely to result in a CONNECTION_BROKEN than with MODE(QUIESCE).

The Secure Sockets Layer (SSL) on WebSphere MQ clients

This chapter discusses the following aspects of using the Secure Sockets Layer (SSL) on WebSphere MQ client systems:

- "Specifying that an MQI channel uses SSL"
- "Specifying the location of LDAP servers that hold certificate revocation lists (CRLs)" on page 101
- "Renegotiating the secret key" on page 103
- "Refreshing a client's view of the SSL key repository contents and SSL settings" on page 103
- "Specifying that only FIPS-certified cryptography will be used" on page 104

For more information about how to implement the SSL support within WebSphere MQ, see WebSphere MQ Security.

Specifying that an MQI channel uses SSL

For an MQI channel to use SSL, the value of the *SSLCipherSpec* attribute of the client-connection channel must be the name of a CipherSpec that is supported by WebSphere MQ on the client platform.

You can define a client-connection channel with a value for this attribute in the following ways. They are listed in order of decreasing precedence.

1. When a WebSphere MQ client application issues an MQCONNX call.

The application can specify the name of a CipherSpec in the *SSLCipherSpec* field of a channel definition structure, MQCD. This structure is referenced by the connect options structure, MQCNO, which is a parameter on the MQCONNX call.

2. Using a client channel definition table.

   One or more entries in a client channel definition table can specify the name of a CipherSpec. For example, if you create an entry by using the DEFINE CHANNEL MQSC command, you can use the SSLCIPH parameter on the command to specify the name of a CipherSpec.
3. Using Active Directory on Windows.

   On Windows systems, you can use the **setmqscp** control command to publish the client-connection channel definitions in Active Directory. One or more of these definitions can specify the name of a CipherSpec.

For example, if a client application provides a client-connection channel definition in an MQCD structure on an MQCONNX call, this definition is used in preference to any entries in a client channel definition table that can be accessed by the WebSphere MQ client.

Note that you cannot use the MQSERVER environment variable to provide the channel definition at the client end of an MQI channel that uses SSL.

To check whether a client certificate has flowed, display the channel status at the server end of a channel for the presence of a peer name parameter value.

Specifying the location of LDAP servers that hold certificate revocation lists (CRLs)

On a WebSphere MQ client system, you can specify the location of Lightweight Directory Access Protocol (LDAP) servers that hold certificate revocation lists (CRLs) in the following ways.

They are listed in order of decreasing precedence.

1. "When a WebSphere MQ client application issues an MQCONNX call"
2. "Using a client channel definition table" on page 102
3. "Using Active Directory on Windows" on page 103

See the relevant sections for more information about each of these ways.

The intention is that each LDAP server holds the same CRLs. The reason for configuring more than one LDAP server with CRLs is to provide higher availability. If one LDAP server is not available when it is required, a WebSphere MQ client can attempt to access another.

When a WebSphere MQ client application issues an MQCONNX call

On an MQCONNX call, the connect options structure, MQCNO, can reference an SSL configuration options structure, MQSCO.

In turn, the MQSCO structure can reference one or more authentication information record structures, MQAIR. Each MQAIR structure contains all the information a WebSphere MQ client needs to access an LDAP server that holds CRLs. For example, one of the fields in an MQAIR structure is the host address or IP address of a system on which an LDAP server runs. This address can be followed by an optional port number enclosed in parentheses. The default port number is 389. For more information about the MQAIR structure, see the WebSphere MQ Application Programming Reference.

Using a client channel definition table

On a server queue manager, you can create one or more authentication information objects. The attributes of an authentication object contain all the information that is needed to access an LDAP server that holds CRLs. One of the attributes specifies the host address or IP address of a system on which an LDAP server runs. This address can be followed by an optional port number enclosed in parentheses. The default port number is 389.

To enable a WebSphere MQ client to access LDAP servers that hold CRLs, the attributes of one or more authentication information objects can be included in a client channel definition table. This is done in the following ways:

On the server platforms AIX, HP-UX, Linux, i5/OS, Solaris, and Windows

You can create a namelist that contains the names of one or more authentication information objects. You can then set the queue manager attribute, *SSLCRLNameList*, to the name of this namelist. By doing this, you enable the WebSphere MQ SSL support for the queue manager to access the LDAP servers that hold CRLs.

The attributes of the authentication information objects identified by the namelist are referred to collectively here as the *CRL information*. When you set the queue manager attribute, *SSLCRLNameList*, to the name of the namelist, the CRL information is copied into the client channel definition table associated with the queue manager. If the client channel definition table can be accessed from a client system as a shared file, or if the client channel definition table is then copied to a client system, the WebSphere MQ client on that system can use the CRL information in the client channel definition table to access LDAP servers that hold CRLs.

If the CRL information of the queue manager is changed subsequently, the change is reflected in the client channel definition table associated with the queue manager. If the queue manager attribute, *SSLCRLNameList*, is set to blank, all the CRL information is removed from the client channel definition table. These changes are not reflected in any copy of the table on a client system.

If you require the CRL information at the client and server ends of an MQI channel to be different, and the server queue manager is the one that is used to create the CRL information, you can do the following:

1. On the server queue manager, create the CRL information for use on the client system.
2. Copy the client channel definition table containing the CRL information to the client system.
3. On the server queue manager, change the CRL information to what is required at the server end of the MQI channel.

On the server platform z/OS

On z/OS, a client channel definition table is generated by the MAKECLNT parameter of the COMMAND function of the WebSphere MQ utility program, CSQUTIL. The DISPLAY CHANNEL commands in the input data set determine which client-connection channel definitions are included in the table. Likewise, the DISPLAY AUTHINFO commands in the input data set determine which authentication information objects are used to form the CRL information in the table.

The contents of a client channel definition table generated on z/OS do not depend on the value of any queue manager attributes, such as *SSLCRLNameList*, and cannot be updated dynamically. The only way you can change the CRL information in a client channel definition table is to generate a new table by running CSQUTIL again.

Using Active Directory on Windows

On Windows systems, you can use the **setmqcrl** control command to publish the current CRL information in Active Directory.

For information about this command and its syntax, see the WebSphere MQ System Administration Guide.

Renegotiating the secret key

During an SSL handshake a 'secret key' is generated to encrypt data between the SSL client and SSL server. The key can be renegotiated periodically to minimize the amount of encrypted data that can be decrypted if the secret key is discovered.

By default, clients do not renegotiate the SSL secret key. You can make a client renegotiate the key by using the KeyResetCount field in one of three ways:

- Setting the SSLKeyResetCount attribute in the client configuration file.
- Using the MQSCO structure on an MQCONNX call
- using the environment variable MQSSLRESET

These variables can be set to an integer in the range 0 through 999 999 999, representing the number of unencrypted bytes sent and received within an SSL conversation before the SSL secret key is renegotiated. Specifying a value of 0 indicates that SSL secret keys are never renegotiated. If you specify an SSL/TLS secret key reset count between 1 byte and 32Kb, SSL/TLS channels will use a secret key reset count of 32Kb. This is to avoid the overhead of excessive key resets which would occur for small SSL/TLS secret key reset values. For an MQCONNX call, if both MQSSLRESET and KeyResetCount are specified, the value of the latter is used.

If a value greater than zero is specified and channel heartbeats are enabled for the channel, the secret key is also renegotiated before message data is sent or received following a channel heartbeat.

The count of bytes until the next secret key renegotiation is reset after each successful renegotiation.

For full details of the MQCONNX function call and the MQSCO sctructure, see WebSphere MQ Application Programming Reference. For full details of MQSSLRESET, see "MQSSLRESET" on page 117.

Refreshing a client's view of the SSL key repository contents and SSL settings

If changes are made on a client machine to the contents or location of the SSL key repository, the authentication information, the cryptographic hardware parameters, or the FIPS-certified cryptography setting, all the SSL connections must be ended. Once they are all ended they can be restarted and will reflect the changes made. A client application, especially a long running application, should therefore be written in such a way that its SSL connections can be stopped and restarted without forcibly breaking the connection to the queue manager. However, if this cannot be done, an application can be forcibly closed. In either case, when the SSL channels restart all the new SSL settings will be used. These settings are the same as those refreshed by the REFRESH SECURITY TYPE(SSL) command (see WebSphere MQ Script (MQSC) Command Reference).

Specifying that only FIPS-certified cryptography will be used

There are three ways to specify that an SSL channel must only use FIPS-certified cryptography.

You can set the SSLFipsRequired attribute in the client configuration file to YES, the environment variable MQSSLFIPS to YES, or the FipsRequired field in the MQSCO structure to MQSSL_FIPS_YES (by default, FIPS-certified cryptography is not required).

These values have the same meanings as they do on ALTER QMGR SSLFIPS (see *WebSphere MQ Script (MQSC) Command Reference*). If the client process currently has no active SSL connections and a FipsRequired value is validly specified on an SSL MQCONNX, all subsequent SSL connections associated with this process must use only the CipherSpecs associated with this value. This applies until this and all other SSL connections have stopped, at which stage a subsequent MQCONNX can provide a new value for FipsRequired.

If cryptographic hardware is configured, the cryptographic modules used are those provided by the hardware product, and these may, or may not, be FIPS-certified to a particular level. This depends on the hardware product in use.

If the MQSSLFIPS and FipsRequired variables are both set but with inconsistent values, the FipsRequired value takes precedence.

If FIPS-only cryptography is specified in this way but the platform is not a FIPS-certified platform, all SSL connections fail with MQRC_SSL_INITIALIZATION_ERROR.

If FIPS-only cryptography is specified in this way and a non-FIPS CipherSpec is specified for a connection, the connection fails with MQRC_SSL_INITIALIZATION_ERROR.

For full details of MQSSLFIPS, see "MQSSLFIPS" on page 116.

WebSphere MQ client configuration file

Configure your clients using attributes in a text file. These attributes can be overridden by environment variables or in other platform-specific ways.

You configure your WebSphere MQ clients using a text file, similar to the queue manager configuration file, qm.ini, used on UNIX platforms. The file contains a number of stanzas, each of which contains a number of lines of the format **attribute-name**=*value*.

In this documentation, this file is referred to as the *WebSphere MQ client configuration file*; its filename is generally mqclient.ini, but you can choose to give it another name. Configuration information in this file applies to all platforms, and to clients using the MQI, WebSphere MQ classes for Java, WebSphere MQ classes for JMS, WebSphere MQ classes for .NET, and XMS.

The configuration features apply to all connections a client application makes to any queue managers, rather than being specific to an individual connection to a queue manager. Attributes relating to a connection to an individual queue manager can be configured programmatically, for example by using an MQCD structure, or by using a Client Channel Definition Table (CCDT).

Environment variables which were supported in releases of WebSphere MQ earlier than Version 7.0 continue to be supported, and where such an environment variable matches an equivalent value in the client configuration file, the environment variable overrides the client configuration file value.

For a client application using WebSphere MQ classes for JMS, you can also override the client configuration file in the following ways:
- setting properties in the JMS configuration file
- setting Java system properties, which also overrides the JMS configuration file

For the .NET client, you can also override the client configuration file and the equivalent environment variables using the .NET Application Configuration File.

Example client configuration file

```
#* Module Name: mqclient.ini                                       *#
#* Type       : WebSphere MQ client configuration file             *#
#  Function   : Define the configuration of a client               *#
#*                                                                 *#
#******************************************************************#
#* Notes      :                                                    *#
#* 1) This file defines the configuration of a client              *#
#*                                                                 *#
#******************************************************************#

ClientExitPath:
   ExitsDefaultPath=/var/mqm/exits
   ExitsDefaultPath64=/var/mqm/exits64

TCP:
   Library1=DLLName1
   KeepAlive = Yes
   ClntSndBuffSize=32768
   ClntRcvBuffSize=32768
   Connect_Timeout=0

MessageBuffer:
   MaximumSize=-1
   Updatepercentage=-1
   PurgeTime=0

LU62:
   TPName
   Library1=DLLName1
   Library2=DLLName2
```

Location of the client configuration file

A WebSphere MQ client configuration file can be held in a number of locations.

A client application uses the following search path to locate the WebSphere MQ client configuration file:
1. The location specified by the environment variable MQCLNTCF.

The format of this environment variable is a full URL. This means the file name might not necessarily be `mqclient.ini` and facilitates placing the file on a network attached file-system.

Note the following:

- C, .NET and XMS clients support only the `file:` protocol; the `file:` protocol is assumed if the URL string does not begin with `protocol:`
- To allow for Java 1.4.2 JREs, which do not support reading environment variables, the MQCLNTCF environment variable can be overridden with an MQCLNTCF Java System Property.

2. A file called `mqclient.ini` in the present working directory of the application.
3. A file called `mqclient.ini` in the WebSphere MQ data directory for UNIX systems or installation directory for Windows.

   Note the following:

   - The WebSphere MQ data directory does not exist on certain platforms, for example, i5/OS and z/OS, or in cases where the client has been supplied with another product.
   - On UNIX systems, the directory is `/var/mqm`
   - On Windows platforms you configure the environment variable MQ_FILE_PATH during installation, to point at the installation directory. It is normally C:\Program Files\IBM\WebSphere MQ
   - To allow for Java 1.4.2 JREs that do not support reading environment variables you can manually override the MQ_FILE_PATH environment variable with an MQ_FILE_PATH Java System Property.
4. A file called `mqclient.ini` in a standard directory appropriate to the platform, and accessible to users:
   - For all Java clients this is the value of the `user.home` Java System Property.
   - For C clients on UNIX platforms this is the value of the HOME environment variable.
   - For C clients on Windows this is the concatenated values of the HOMEDRIVE and HOMEPATH environment variables.

CHANNELS stanza of the client configuration file

Use the CHANNELS stanza to specify information about client channels.

The following attributes can be included in the CHANNELS stanza:

CCSID=*number*
: The coded character set number to be used.

  This is equivalent to the MQCCSID environment parameter.

ChannelDefinitionDirectory=*path*
: The directory path to the file containing the client channel definition table.

  On Windows the default is the WebSphere MQ installation directory, normally C:\Program Files\IBM\WebSphere MQ: on Unix systems the default is /var/mqm.

  This is equivalent to the MQCHLLIB environment parameter.

ChannelDefinitionFile=*filename* | **AMQCLCHL.TAB**
: The name of the file containing the client channel definition table.

  This is equivalent to the MQCHLTAB environment parameter.

ServerConnectionParms

ServerConnectionParms specifies the location of the WebSphere MQ server and the communication method to be used. This attribute defines only a simple channel; you cannot use it to define an SSL channel or a channel with channel exits. It is a string of the format *ChannelName/TransportType/ConnectionName*. *ConnectionName* must be a fully-qualified network name. *ChannelName* cannot contain the forward slash (/) character because this character is used to separate the channel name, transport type, and connection name. When ServerConnectionParms is used to define a client channel, a maximum message length of 100 MB is used. Therefore the maximum message size in effect for the channel is the value specified in the SVRCONN channel at the server.

This is equivalent to the MQSERVER environment parameter.

ClientExitPath stanza of the client configuration file

Use the ClientExitPath stanza to specify the default locations of channel exits on the client.

The following attributes can be included in the ClientExitPath stanza:

ExitsDefaultPath=*string*

Specifies the location of 32-bit channel exits for clients.

ExitsDefaultPath64=*string*

Specifies the location of 64-bit channel exits for clients.

JavaExitsClassPath=*string*

The values to be added to the classpath when a Java exit is run. This is ignored by exits in any other language.

In the JMS configuration file, the JavaExitsClassPath name is given the standard com.ibm.mq.cfg. prefix and this full name is also used on the Websphere MQ V7.0 system property. At Version 6.0 this attribute was specified using system property com.ibm.mq.exitClasspath, which was documented in the Version 6.0 readme. The use of com.ibm.mq.exitClasspath is deprecated. If both JavaExitsClassPath and exitClasspath are present, JavaExitsClassPath is honored. If only exitClasspath usage is present, it is still honored in Websphere MQ V7.0.

LU62, NETBIOS, and SPX stanzas of the client configuration file

On Windows systems only, use these stanzas to specify configuration parameters for the specified network protocols.

LU62

Use the LU62 stanza to specify SNA LU 6.2 protocol configuration parameters. The following attributes can be included in this stanza:

Library1=*DLLName***|WCPIC32**

The name of the APPC DLL.

Library2=*DLLName***|WCPIC32**

The same as Library1, used if the code is stored in two separate libraries. .

TPName

The TP name to start on the remote site.

NETBIOS

Use the NETBIOS stanza to specify NetBIOS protocol configuration parameters. The following attributes can be included in this stanza:

AdapterNum=*number***|****0**
: The number of the LAN adapter.

Library1=*DLLName***|****NETAPI32**
: The name of the NetBIOS DLL.

LocalName=*name*
: The name by which this computer is known on the LAN.

  This is equivalent to the MQNAME environment parameter.

NumCmds=*number***|****1**
: The number of commands to allocate.

NumSess=*number***|****1**
: The number of sessions to allocate.

SPX

Use the SPX stanza to specify SPX protocol configuration parameters. The following attributes can be included in this stanza:

BoardNum=*number***|****0**
: The LAN adapter number.

KeepAlive=YES|NO
: Switch the KeepAlive function on or off.

  KeepAlive=YES causes SPX to check periodically that the other end of the connection is still available. If it is not, the channel is closed.

Library1=*DLLName***|****WSOCK32.DLL**
: The name of the SPX DLL.

Library2=*DLLName***|****WSOCK32.DLL**
: The same as Library1, used if the code is stored in two separate libraries.

Socket=*number***|****5E86**
: The SPX socket number in hexadecimal notation.

MessageBuffer stanza of the client configuration file

Use the MessageBuffer stanza to specify information about message buffers.

The following attributes can be included in the MessageBuffer stanza:

MaximumSize=*integer***|****1**
: Size, in kilobytes, of the read-ahead buffer, in the range 1 - 999 999.

  The following special values exist:

  -1 The client determines the appropriate value.

  0 Read ahead is disabled for the client.

PurgeTime=*integer***|****600**
: Interval, in seconds, after which messages left in the read-ahead buffer are purged.

If the client application is selecting messages based on MsgId or CorrelId it is possible that the read ahead buffer could contain messages sent to the client with a previously requested MsgId or CorrelId. These messages would then be stranded in the read ahead buffer until an MQGET is issued with an appropriate MsgId or CorrelId. You can purge messages from the read ahead buffer by setting PurgeTime. Any messages that have remained in the read ahead buffer for longer than the purge interval are automatically purged. These messages have already been removed from the queue on the queue manager, so unless they are being browsed, they will be lost.

The valid range is 1 - 999 999 seconds, or the special value 0, meaning that no purge takes place.

UpdatePercentage=*integer* **|** **-1**

The update percentage value, in the range of 1 - 100, used in calculating the threshold value to determine when a client application makes a new request to the server. The special value -1 indicates that the client determines the appropriate value.

The client periodically sends a request to the server indicating how much data the client application has consumed. A request is sent when the number of bytes, *n*, retrieved by the client via MQGET calls exceeds a threshold *T*. *n* is reset to zero each time a new request is sent to the server.

The threshold T is calculated as follows:

```
T = Upper - Lower
```

Upper is the same as the read-ahead buffer size, specified by the *MaximumSize* attribute, in Kilobytes. Its default is 100 Kb.

| Lower is lower than Upper, and is specified by the *UpdatePercentage* attribute.
| This attribute is a number between 1 and 100, and has a default of 20. Lower is
| calculated as follows:

```
Lower = Upper x UpdatePercentage / 100
```

Example 1:

The MaximumSize and UpdatePercentage attributes take their defaults of 100 Kb and 20, respectively.

The client calls MQGET to retrieve a message, and does so repeatedly. This continues until MQGET has consumed n bytes.

Using the calculation above, T is (100 - 20) = 80 Kb.

So when MQGET calls have removed 80 Kb from a queue, the client makes a new request automatically.

Example 2:

The MaximumSize attributes takes its default of 100 Kb, and a value of 40 is chosen for UpdatePercentage.

The client calls MQGET to retrieve a message, and does so repeatedly. This continues until MQGET has consumed n bytes.

Using the calculation above, T is (100 - 40) = 60 Kb

So when MQGET calls have removed 60 Kb from a queue, the client makes a new request automatically. This is sooner than in EXAMPLE 1 where the defaults were used.

Therefore choosing a larger threshold *T* will tend to decrease the frequency at which requests are sent from client to server. Conversely choosing a smaller threshold *T* will tend to increase the frequency of requests that are sent from client to server.

However, choosing a large threshold *T* can mean that the performance gain of read ahead is reduced as the chance of the read ahead buffer becoming empty can increase. When this happens an MQGET call might have to pause, waiting for data to arrive from the server.

SSL stanza of the client configuration file

Use the SSL stanza to specify information about the use of SSL.

The following attributes can be included in the SSL stanza:

SSLCryptoHardware=*string*
Sets the name of the parameter string required to configure the cryptographic hardware present on the system.

Specify a string of the following format: **GSK_PKCS11**=*driver path and filename*;*token label*;*token password*;*symmetric cipher setting*;

The driver path is an absolute path to the shared library providing support for the PKCS #11 card. The driver filename is the name of the shared library. An example of the value required for the PKCS #11 driver path and filename is /usr/lib/pkcs11/PKCS11_API.so To access symmetric cipher operations through GSKit, specify the symmetric cipher setting parameter. The value of this parameter is either:

SYMMETRIC_CIPHER_OFF
Do not access symmetric cipher operations. This is the default setting.

SYMMETRIC_CIPHER_ON
Access symmetric cipher operations.

The maximum length of the string is 256 characters. The default value is blank. If you specify a string that is not in the format listed above, you get an error.

SSLFipsRequired=YES | NO
Specifies whether only FIPS-certified algorithms are to be used if cryptography is carried out in WebSphere MQ. If cryptographic hardware is configured, the cryptographic modules used are those provided by the hardware product, and these might, or might not, be FIPS-certified to a particular level. This depends on the hardware product in use.

SSLKeyRepository=*pathname*
The location of the key repository that holds the user's digital certificate, in stem format. That is, it includes the full path and the file name without an extension.

SSLKeyResetCount=*integer* **| 0**
The number of unencrypted bytes sent and received on an SSL channel before the SSL secret key is renegotiated.

The value must be in the range 0 - 999 999 999.

The default is 0, which means that secret keys are never renegotiated.

If you specify a value between 1 and 32768, SSL and TLS channels will use a secret key reset count of 32768 (32Kb). This is to avoid the overhead of excessive key resets which would occur for small secret key reset values.

TCP stanza of the client configuration file

Use the TCP stanza to specify TCP network protocol configuration parameters.

The following attributes can be included in the TCP stanza:

ClntRcvBuffSize=*number***|**<u>**32768**</u>
: The size in bytes of the TCP/IP receive buffer.

ClntSndBuffSize=*number***|**<u>**32768**</u>
: The size in bytes of the TCP/IP send buffer.

Connect_Timeout=*number*
: The number of seconds before an attempt to connect the socket will timeout; the default is 0 unless the channel has been configured with a non-zero client channel weighting in which case the default is 5.

IPAddressVersion=<u>**MQIPADDR_IPV4**</u>**|MQIPADDR_IPV6**
: Specifies which IP protocol to use for a channel connection.

  It has the possible string values of MQIPADDR_IPV4 or MQIPADDR_IPV6. These values have the same meanings as IPV4 and IPV6, respectively, in ALTER QMGR IPADDRV.

KeepAlive=YES|<u>**NO**</u>
: Switch the KeepAlive function on or off. KeepAlive=YES causes TCP/IP to check periodically that the other end of the connection is still available. If it is not, the channel is closed.

Library1=*DLLName***|**<u>**WSOCK32**</u>
: (Windows only) The name of the TCP/IP sockets DLL.

Using WebSphere MQ environment variables

This chapter describes the environment variables that you can use with WebSphere MQ client applications.

WebSphere MQ uses default values for those variables that you have not set. Using environment variables, you can update your system profile to make a permanent change, issue the command from the command line to make a change for this session only, or, if you want one or more variables to have a particular value dependent on the application that is running, add commands to a command script file used by the application.

The WebSphere MQ environment variables are:
- "MQCCSID" on page 112
- "MQCHLLIB" on page 112
- "MQCHLTAB" on page 113
- "MQIPADDRV" on page 113
- "MQNAME" on page 114
- "MQSERVER" on page 114
- "MQSSLCRYP" on page 116
- "MQSSLFIPS" on page 116
- "MQSSLKEYR" on page 116
- "MQSSLRESET" on page 117

Commands are available on all the WebSphere MQ client platforms unless otherwise stated.

Note:

1. WebSphere MQ for z/OS does not support any WebSphere MQ environment variables. If you are using this platform as your server, see "Client channel definition table" on page 94 for information about how the client channel definition table is generated on z/OS. You can still use the WebSphere MQ environment variables on your client platform.

For each environment variable, use the command relevant to your platform to display the current setting or to reset the value of a variable.

For example:

Command	**Effect**
SET MQSERVER=	Removes the variable from Windows environments
unset MQSERVER	Removes the variable from UNIX systems environments
SET MQSERVER	Displays the current setting on Windows
echo $MQSERVER	Displays the current setting on UNIX systems
set	Displays all environment variables for the session

MQCCSID

This specifies the coded character set number to be used and overrides the machine's configured CCSID.

See "Choosing client or server coded character set identifier (CCSID)" on page 120 for more information.

To set this variable use one of these commands:

- For Windows:
```
SET MQCCSID=number
```
- For UNIX systems:
```
export MQCCSID=number
```

For more information, see the WebSphere MQ Application Programming Reference manual.

MQCHLLIB

This specifies the directory path to the file containing the client channel definition table.

The file is created on the server, but can be copied across to the WebSphere MQ client machine. If MQCHLLIB is not set, the path defaults to:

- For Windows :
```
mqmtop
```
- For UNIX systems:
```
/var/mqm/
```

If you are using WebSphere MQ for z/OS as your server, the file must be kept on the WebSphere MQ client machine.

For servers on other platforms, consider keeping this file on the server to make administration easier.

To set this variable use one of these commands:

- For Windows:
  ```
  SET MQCHLLIB=pathname
  ```
- For UNIX systems:
  ```
  export MQCHLLIB=pathname
  ```

For example:

```
SET MQCHLLIB=C:\wmqtest
```

Note:

1. If you change the setting of this variable ***after*** you create the queue manager, you must copy your existing client channel definition table to the new location.
2. If you change the setting of this variable ***before*** you create the queue manager, you do not need to copy your existing client channel definition table to the new location.

MQCHLTAB

This specifies the name of the file containing the client channel definition table.

The default file name is AMQCLCHL.TAB. For information about where the client channel definition table is located on a server machine, see "Client channel definition table" on page 94.

To set this variable use one of these commands:

- On Windows:
  ```
  SET MQCHLTAB=filename
  ```
- On UNIX systems:
  ```
  export MQCHLTAB=filename
  ```

For example:

```
SET MQCHLTAB=ccdf1.tab
```

Using MQCHLLIB and MQCHLTAB on the server

In the same way as for the client, the MQCHLLIB environment variable on the server specifies the path to the directory containing the client channel definition table. Similarly, the MQCHLTAB environment variable on the server specifies the name of the client channel definition table.

MQIPADDRV

This specifies which IP protocol to use for a channel connection.

It has the possible string values of ″MQIPADDR_IPV4″ or ″MQIPADDR_IPV6″. These values have the same meanings as IPV4 and IPV6, respectively, in ALTER QMGR IPADDRV.

If it is not set, ″MQIPADDR_IPV4″ is assumed.

To set this variable use one of these commands:

- For Windows:

```
SET MQIPADDRV=MQIPADDR_IPV4|MQIPADDR_IPV6
```

- For UNIX systems:

  ```
  export MQIPADDRV=MQIPADDR_IPV4|MQIPADDR_IPV6
  ```

For more information, see the WebSphere MQ Application Programming Reference manual.

MQNAME

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

See "Defining a NetBIOS connection" on page 64 for a full description and for the rules of precedence on the client and the server.

To set this variable use this command:

```
SET MQNAME=Your_env_Name
```

For example:

```
SET MQNAME=CLIENT1
```

The NetBIOS on some platforms requires a different name (set by MQNAME) for each application if you are running multiple WebSphere MQ applications simultaneously on the WebSphere MQ client.

MQSERVER

This environment variable is used to define a minimal channel.

| You cannot use MQSERVER to define an SSL channel or a channel with channel exits. MQSERVER specifies the location of the WebSphere MQ server and the communication method to be used. *ConnectionName* must be a fully-qualified network name. The *ChannelName* cannot contain the forward slash (/) character because this character is used to separate the channel name, transport type, and connection name. When the MQSERVER environment variable is used to define a client channel, a maximum message length (MAXMSGL) of 100 MB is used. Therefore the maximum message size in effect for the channel is the value specified in the SVRCONN channel at the server.

To set this variable use one of these commands:

- For Windows:

  ```
  SET MQSERVER=ChannelName/TransportType/ConnectionName
  ```

- For UNIX systems:

  ```
  export MQSERVER=ChannelName/TransportType/ConnectionName
  ```

TCP/IP default port

By default, for TCP/IP, WebSphere MQ assumes that the channel will be connected to port 1414.

You can change this by:

- Adding the port number in brackets as the last part of the ConnectionName:
  - For Windows:

    ```
    SET MQSERVER=ChannelName/TransportType/ConnectionName(PortNumber)
    ```

  - For UNIX systems:

    ```
    export MQSERVER=ChannelName/TransportType/ConnectionName(PortNumber)
    ```

|

- Changing the `mqclient.ini` file by adding the port number to the protocol name, for example:

  ```
  TCP:
  port=2001
  ```
- Adding WebSphere MQ to the services file as described in “Defining a TCP/IP connection” on page 63.

SPX default socket

By default, for SPX, WebSphere MQ assumes that the channel will be connected to socket 5E86.

You can change this by:

- Adding the socket number in brackets as the last part of the ConnectionName:

  ```
  SET MQSERVER=ChannelName/TransportType/ConnectionName(SocketNumber)
  ```

  For SPX connections, specify the ConnectionName and socket in the form `network.node(socket)`. If the WebSphere MQ client and server are on the same network, the network need not be specified. If you are using the default socket, the socket need not be specified.
- Changing the `qm.ini` file by adding the port number to the protocol name, for example:

  ```
  SPX:
  socket=5E87
  ```

Using MQSERVER

If you use the MQSERVER environment variable to define the channel between your WebSphere MQ client machine and a server machine, this is the only channel available to your application, and no reference is made to the client channel definition table.

In this situation, the listener program that you have running on the server machine determines the queue manager to which your application will connect. It will be the same queue manager as the listener program is connected to.

If the MQCONN or MQCONNX request specifies a queue manager other than the one the listener is connected to, or if the MQSERVER parameter *TransportType* is not recognized, the MQCONN or MQCONNX request fails with return code MQRC_Q_MGR_NAME_ERROR.

Example of using MQSERVER:

Examples on a UNIX system:

```
export MQSERVER=CHANNEL1/TCP/'9.20.4.56(2002)'
export MQSERVER=CHANNEL1/LU62/BOX99
```

All **MQCONN** or **MQCONNX** requests then attempt to use the channel you have defined unless an MQCD structure has been referenced from the MQCNO structure supplied to **MQCONNX**, in which case the channel specified by the MQCD structure takes priority over any specified by the MQSERVER environment variable.

The MQSERVER environment variable takes priority over any client channel definition pointed to by MQCHLLIB and MQCHLTAB.

Canceling MQSERVER

To cancel MQSERVER and return to the client channel definition table pointed to by MQCHLLIB and MQCHLTAB, enter the following:

- On Windows:
  ```
  SET MQSERVER=
  ```
- On UNIX systems:
  ```
  unset MQSERVER
  ```

MQSSLCRYP

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

To set this variable use one of these commands:

- On Windows systems:
  ```
  SET MQSSLCRYP=string
  ```
- On UNIX systems:
  ```
  export MQSSLCRYP=string
  ```

MQSSLFIPS

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

This is affected by the use of cryptographic hardware, see “Specifying that only FIPS-certified cryptography will be used” on page 104.

To set this variable use one of these commands:

- On Windows systems:
  ```
  SET MQSSLFIPS=YES|NO
  ```
- On UNIX systems:
  ```
  export MQSSLFIPS=YES|NO
  ```

The default is `NO`.

MQSSLKEYR

This variable specifies the location of the key repository that holds the user’s digital certificate, in stem format. That is, it includes the full path and the file name without an extension. For full details, see the SSLKEYR parameter of the ALTER QMGR command.

To set this variable use one of these commands:

- On Windows systems:
  ```
  SET MQSSLKEYR=pathname
  ```
- On UNIX systems:
  ```
  export MQSSLKEYR=pathname
  ```

There is no default value.

MQSSLRESET

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

See "Renegotiating the secret key" on page 103 for more information about secret key renegotiation.

It can be set to an integer in the range 0 through 999 999 999. The default is 0, which indicates that SSL secret keys are never renegotiated. If you specify an SSL/TLS secret key reset count between 1 byte and 32Kb, SSL/TLS channels will use a secret key reset count of 32Kb. This is to avoid the overhead of excessive key resets which would occur for small SSL/TLS secret key reset values.

To set this variable use one of these commands:

- On Windows systems:
  ```
  SET MQSSLRESET=integer
  ```
- On UNIX systems:
  ```
  export MQSSLRESET=integer
  ```

Chapter 4. Application programming

Using the message queue interface (MQI)

When you write your WebSphere MQ application, you need to be aware of the differences between running it in a WebSphere MQ client environment and running it in the full WebSphere MQ queue manager environment.

This chapter discusses the following topics:
- "Limiting the size of a message"
- "Choosing client or server coded character set identifier (CCSID)" on page 120
- "Designing applications" on page 120
- "Using MQINQ" on page 120
- "Using syncpoint coordination" on page 121
- "Using MQCONNX" on page 124

Limiting the size of a message

The maximum message length (MaxMsgLength) attribute of a queue manager is the maximum length of a message that can be handled by that queue manager. The default maximum message length supported depends on the platform you are using.

On WebSphere MQ products, you can increase the maximum message length attribute of a queue manager. Details are given in the WebSphere MQ Application Programming Guide.

You can find out the value of MaxMsgLength for a queue manager by using the MQINQ call.

If the MaxMsgLength attribute is changed, no check is made that there are not already queues, and even messages, with a length greater than the new value. After a change to this attribute, applications and channels should be restarted in order to ensure that the change has taken effect. It will then not be possible for any new messages to be generated that exceed either the queue manager's MaxMsgLength or the queue's MaxMsgLength (unless queue manager segmentation is allowed).

The maximum message length in a channel definition limits the size of a message that you can transmit along a client connection. If a WebSphere MQ application tries to use the MQPUT call or the MQGET call with a message larger than this, an error code is returned to the application. The maximum message size parameter of the channel definition does not affect the maximum message size which can be consumed using MQCB over a client connection.

Choosing client or server coded character set identifier (CCSID)

The data passed across the MQI from the application to the client stub should be in the local coded character set identifier (CCSID), encoded for the WebSphere MQ client. If the connected queue manager requires the data to be converted, then this is done by the queue manager.

The Java client in V7, however, is capable of doing the conversion if the queue manager is unable to do so. Please see the *Client connections* section in *WebSphere MQ Using Java*.

The client code assumes that the character data crossing the MQI in the client is in the CCSID configured for that machine. If this CCSID is an unsupported CCSID or is not the required CCSID, it can be overridden with the MQCCSID environment variable, for example on Windows:

```
SET MQCCSID=850
```

Or, on UNIX or Linux systems: `export MQCCSID=850`

Set this in the profile and all MQI data will be assumed to be in code page 850.

Note: This does not apply to application data in the message.

CCSID and encoding fields - multiple puts

If your application is performing multiple PUTs that include WebSphere MQ headers after the message descriptor (MQMD), be aware that the CCSID and encoding fields of the MQMD are overwritten after completion of the first PUT.

After the first PUT, these fields contain the value used by the connected queue manager to convert the WebSphere MQ headers. Ensure that your application resets the values to those it requires.

Designing applications

When designing an application, consider what controls you need to impose during an MQI call to ensure that the WebSphere MQ application processing is not disrupted.

Using MQINQ

Some values queried using MQINQ are modified by the client code.

CCSID
: is set to the client CCSID, not that of the queue manager.

MaxMsgLength
: is reduced if it is restricted by the channel definition. This will be the lower of:
  - The value defined in the queue definition, or
  - The value defined in the channel definition

For more information, see the WebSphere MQ Application Programming Guide.

Using syncpoint coordination

Within WebSphere MQ, one of the roles of the queue manager is syncpoint control within an application. If an application runs on a WebSphere MQ base client, it can issue MQCMIT and MQBACK, but the scope of the syncpoint control is limited to the MQI resources. The WebSphere MQ verb MQBEGIN is not valid in a base client environment.

Applications running in the full queue manager environment on the server can coordinate multiple resources (for example databases) via a transaction monitor. On the server you can use the Transaction Monitor supplied with WebSphere MQ products, or another transaction monitor such as CICS. You cannot use a transaction monitor with a base client application.

You can use an external transaction manager with a WebSphere MQ extended transactional client. See “What is an extended transactional client?” on page 3 for details.

Using read ahead

You can use read ahead on a client to allow non persistent messages to be sent to a client without the client application having to request the messages.

When a client requires a message from a server, it sends a request to the server. It sends a separate request for each of the messages it consumes. To improve the performance of a client consuming non persistent messages by avoiding having to send these request messages, a client can be configured to use read ahead. Read ahead allows messages to be sent to a client without an application having to request them.

Using read ahead can improve performance when consuming non persistent messages from a client application. This performance improvement is available to both MQI and JMS applications. Client applications using MQGET or asynchronous consume will benefit from the performance improvements when consuming non-persistent messages.

When read ahead is enabled, messages are sent to an in memory buffer on the client called the read ahead buffer. The client will have a read ahead buffer for each queue it has open with read ahead enabled. The messages in the read ahead buffer are not persisted. The client periodically updates the server with information about the amount of data it has consumed.

Not all client application designs are suited to using read ahead as not all options are supported for use with read ahead and some options are required to be consistent between MQGET calls when read ahead is enabled. If a client alters its selection criteria between MQGET calls, messages being stored in the read ahead buffer will remain stranded in the client read ahead buffer. For more information, see Improving performance of non-persistent messages

Read ahead configuration is controlled by three attributes, MaximumSize, PurgeTime, and UpdatePercentage, which are specified in the MessageBuffer stanza of the WebSphere MQ client configuration file.

Using asynchronous put

Using asynchronous put, an application can put a message to a queue without waiting for a response from the queue manager. You can use this to improve messaging performance in some situations.

Normally, when an application puts a message or messages on a queue, using MQPUT or MQPUT1, the application has to wait for the queue manager to confirm that it has processed the MQI request. You can improve messaging performance, particularly for applications that use client bindings, and applications that put large numbers of small messages to a queue, by choosing instead to put messages asynchronously. When an application puts a message asynchronously, the queue manager does not return the success or failure of each call, but you can instead check for errors periodically.

To put a message on a queue asynchronously, use the MQPMO_ASYNC_RESPONSE option in the `Options` field of the MQPMO structure.

If a message is not eligible for asynchronous put, it is put to a queue synchronously.

When requesting asynchronous put response for MQPUT or MQPUT1, a CompCode and Reason of MQCC_OK and MQRC_NONE does not necessarily mean that the message was successfully put to a queue. Although the success or failure of each individual MQPUT or MQPUT1 call may not be returned immediately, the first error that occurred under an asynchronous call can be determined later through a call to MQSTAT.

For more details on MQPMO_ASYNC_RESPONSE, see *WebSphere MQ Application Programming Reference*.

The Asynchronous Put sample program demonstrates some of the features available. For details of the features and design of the program, and how to run it, see the *WebSphere MQ Application Programming Guide*.

Using sharing conversations

In an environment where sharing conversations is permitted, conversations can share an instance of a client-connection server-connection channel.

Sharing conversations is controlled by two fields, both called SharingConversations, one of which is part of the channel definition (MQCD) structure and one of which is part of the channel exit parameter (MQCXP) structure. The SharingConversations field in the MQCD is an integer value, determining the maximum number of conversations that can share a channel instance associated with the channel. The SharingConversations field in the MQCXP is a boolean value, indicating whether the channel instance is currently shared.

In an environment where sharing conversations is not permitted, new client connections specifying identical MQCDs will not share a channel instance.

A new client application connection will share the channel instance when the following conditions are true:

- Both the client-connection and server-connection ends of the channel instance are configured for sharing conversations, and these values are not overridden by channel exits.
- The client connection MQCD value (supplied on the client MQCONNX call or from the client channel definition table (CCDT)) exactly matches the client connection MQCD value supplied on the client MQCONNX call or from the CCDT when the existing channel instance was first established. Note that the original MQCD might have been subsequently altered by exits or by channel negotiation, but that the match is made against the value which was supplied to the client system before these changes were made.
- The sharing conversations limit on the server side is not exceeded.

If a new client application connection matches the criteria to run sharing a channel instance with other conversations, this decision is made before any exits are called on that conversation. Exits on such a conversation cannot alter the fact that it is sharing the channel instance with other conversations. If there are no existing channel instances matching the new channel definition, a new channel instance is connected.

Channel negotiation only occurs for the first conversation on a channel instance; the negotiated values for the channel instance are fixed at that stage and cannot be altered when subsequent conversations start. TLS/SSL authentication also only occurs for the first conversation.

If the MQCD SharingConversations value is altered during the initialization of any security, send or receive exits for the first conversation on the socket at either the client-connection or the server-connection end of the channel instance, the new value it has after all these exits are initialized is used to determine the sharing conversations value for the channel instance (the lowest value takes precedence).

If the negotiated value for sharing conversations is zero, the channel instance is never shared. Further exit programs that set this field to zero similarly run on their own channel instance.

If the negotiated value for sharing conversations is greater than zero then MQCXP SharingConversations is set to TRUE for subsequent calls to exits, indicating that other exit programs on this channel instance can be entered simultaneously with this one.

When you write a channel exit program, consider whether it will run on a channel instance that might involve sharing conversations. If the channel instance might involve sharing conversations, consider the impact on other instances of the channel exit of changing MQCD fields; all MQCD fields have common values across all the sharing conversations. After the channel instance is established, if exit programs try to alter MQCD fields they might encounter problems because other instances of exit programs running on the channel instance could be attempting to alter the same fields at the same time. If this situation could arise with your exit programs, you must serialize access to the MQCD in your exit code.

If you are working with a channel which is defined to share conversations, but you do not want sharing to occur on a particular channel instance, set the MQCD value of SharingConversations to 1 or 0 when you initialize a channel exit on the first conversation on the channel instance. See SharingConversations for an explanation of the values of SharingConversations.

Example

Sharing conversations is enabled.

You are using a client-connection channel definition which specifies an exit program.

The first time that this channel starts, the exit program alters some of the MQCD parameters when it is initialized. These are acted on by the channel, so the definition that the channel is running with is now different from the one that was originally supplied. The MQCXP SharingConversations parameter is set to TRUE.

The next time that the application connects using this channel, the conversation runs on the channel instance which was started previously, because it has the same original channel definition. The conversation is now running on a channel instance which was set up using the definition altered by the exit on the first conversation. When the exit program is initialized for the second conversation, although it can alter MQCD fields, they are *not* acted on by the channel. These same characteristics apply to any subsequent conversations which share the channel instance.

Using MQCONNX

You can use the MQCONNX call to specify a channel definition (MQCD) structure in the MQCNO structure.

This allows the calling client application to specify the definition of the client-connection channel at run-time. For more information, see “Using the MQCNO structure on an MQCONNX call” on page 92. When you use MQCONNX, the call issued at the server depends on the server level and listener configuration.

When you use MQCONNX from a client, the following options are ignored:
- MQCNO_STANDARD_BINDING
- MQCNO_FASTPATH_BINDING

The MQCD structure you can use depends on the MQCD version number you are using. For information on MQCD versions (MQCD_VERSION), see WebSphere MQ Intercommunication. You can use the MQCD structure, for instance, to pass channel-exit programs to the server. If you are using MQCD Version 3 or later, you can use the structure to pass an array of exits to the server. You can use this function to perform more than one operation on the same message, such as encryption and compression, by adding an exit for each operation, rather than modifying an existing exit. If you do not specify an array in the MQCD structure, the single exit fields will be checked. For more information on channel-exit programs, see WebSphere MQ Intercommunication.

Shared connection handles on MQCONNX

You can share handles between different threads within the same process, using shared connection handles.

When you specify a shared connection handle, the connection handle returned from the MQCONNX call can be passed in subsequent MQI calls on any thread in the process.

Note: You can use a shared connection handle on a WebSphere MQ client to connect to a server queue manager that does not support shared connection handles.

For more information, see the WebSphere MQ Application Programming Reference.

Building applications for WebSphere MQ clients

This chapter lists points to consider when running an application in a WebSphere MQ client environment, and describes how to link your application code with the WebSphere MQ client code.

It discusses the following topics:
- "Running applications in the WebSphere MQ client environment"
- "Triggering in the client environment" on page 126
- "Linking C applications with the WebSphere MQ client code" on page 127
- "Linking C++ applications with the WebSphere MQ client code" on page 128
- "Linking COBOL applications with the WebSphere MQ client code" on page 128
- "Linking Visual Basic applications with the WebSphere MQ client code" on page 129

If an application is to run in a client environment, you can write it in the languages shown in the following table:

Table 19. Programming languages supported in client environments

Client platform	C	C++	COBOL	Visual Basic
AIX	Yes	Yes	Yes	
HP-UX	Yes	Yes	Yes	
Linux	Yes	Yes		
Solaris	Yes	Yes	Yes	
Windows	Yes	Yes	Yes	Yes

Running applications in the WebSphere MQ client environment

You can run a WebSphere MQ application both in a full WebSphere MQ environment and in a WebSphere MQ client environment without changing your code, provided that certain conditions are met.

These conditions are that:
- The application does not need to connect to more than one queue manager concurrently.
- The queue manager name is not prefixed with an asterisk (*) on an **MQCONN** or **MQCONNX** call.
- The application does not need to use any of the exceptions listed in .

Note: The libraries that you use at link-edit time determine the environment in which your application must run.

When working in the WebSphere MQ client environment, remember that:

- Each application running in the WebSphere MQ client environment has its own connections to servers. An application establishes one connection to a server each time it issues an **MQCONN** or **MQCONNX** call.
- An application sends and gets messages synchronously. This implies a wait between the time the call is issued at the client and the return of a completion code and reason code across the network.
- All data conversion is done by the server, but see also "MQCCSID" on page 112 for information about overriding the machine's configured CCSID.

Triggering in the client environment

Triggering is explained in detail in the WebSphere MQ Application Programming Guide.

Messages sent by WebSphere MQ applications running on WebSphere MQ clients contribute to triggering in exactly the same way as any other messages, and they can be used to trigger programs on both the server and the client. The trigger monitor and the application to be started must be on the same system.

The default characteristics of the triggered queue are the same as those in the server environment. In particular, if no MQPMO syncpoint control options are specified in a client application putting messages to a triggered queue that is local to a z/OS queue manager, the messages are put within a unit of work. If the triggering condition is then met, the trigger message is put on the initiation queue within the same unit of work and cannot be retrieved by the trigger monitor until the unit of work ends. The process that is to be triggered is not started until the unit of work ends.

Process definition

You must define the process definition on the server, because this is associated with the queue that has triggering set on.

The process object defines what is to be triggered. If the client and server are not running on the same platform, any processes started by the trigger monitor must define *ApplType*, otherwise the server takes its default definitions (that is, the type of application that is normally associated with the server machine) and causes a failure.

For example, if the trigger monitor is running on a Windows client and wants to send a request to a server on another operating system, MQAT_WINDOWS_NT must be defined otherwise the other operating system uses its default definitions and the process fails.

For a list of application types, see the WebSphere MQ Application Programming Reference manual.

Trigger monitor

The trigger monitor provided by non-z/OS WebSphere MQ products runs in the client environments for UNIX and Windows systems.

To run the trigger monitor, issue the command:

```
runmqtmc [-m QMgrName] [-q InitQ]
```

The default initiation queue is SYSTEM.DEFAULT.INITIATION.QUEUE on the default queue manager. This is where the trigger monitor looks for trigger messages. It then calls programs for the appropriate trigger messages. This trigger monitor supports the default application type and is the same as `runmqtrm` except that it links the client libraries.

The command string, built by the trigger monitor, is as follows:

1. The *ApplicId* from the relevant process definition. This is the name of the program to run, as it would be entered on the command line.
2. The `MQTMC2` structure, enclosed in quotes, as got from the initiation queue. A command string is invoked that has this string, exactly as provided, in quotes in order that the system command will accept it as one parameter.
3. The *EnvrData* from the relevant process definition.

The trigger monitor does not look to see if there is another message on the initiation queue until the completion of the application it has just started. If the application has a lot of processing to do, this might mean that the trigger monitor cannot keep up with the number of trigger messages arriving. There are two ways to deal with this:

1. Have more trigger monitors running

   If you choose to have more trigger monitors running, you can control the maximum number of applications that can run at any one time.
2. Run the started applications in the background

   If you choose to run applications in the background, WebSphere MQ imposes no restriction on the number of applications that can run.

To run the started application in the background on a UNIX system, you must put an & (ampersand) at the end of the *EnvrData* of the process definition.

CICS applications (non-z/OS)

A non-z/OS CICS application program that issues an **MQCONN** or **MQCONNX** call must be defined to CEDA as RESIDENT. To make the resident code as small as possible, you can link to a separate program to issue the **MQCONN** or **MQCONNX** call.

If the MQSERVER environment variable is used to define the client connection, it must be specified in the CICSENV.CMD file.

WebSphere MQ applications can be run in a WebSphere MQ server environment or on a WebSphere MQ client without changing code. However, in a WebSphere MQ server environment, CICS can act as syncpoint coordinator, and you use EXEC CICS SYNCPOINT and EXEC CICS SYNCPOINT ROLLBACK rather than **MQCMIT** and **MQBACK**. If a CICS application is simply relinked as a client, syncpoint support is lost. **MQCMIT** and **MQBACK** must be used for the application running on a WebSphere MQ client.

Linking C applications with the WebSphere MQ client code

Having written your WebSphere MQ application that you want to run on the WebSphere MQ client, you must link it to a queue manager.

You can do this in two ways:

1. Directly, in which case the queue manager must be on the same machine as your application

2. To a client library file, which gives you access to queue managers on the same or on a different machine

WebSphere MQ provides a client library file for each environment:

AIX libmqic.a library for non-threaded applications, or libmqic_r.a library for threaded applications.

HP-UX
libmqic.sl library for non-threaded applications, or libmqic_r.sl library for threaded applications.

Linux libmqic.so library for non-threaded applications, or libmqic_r.so library for threaded applications.

Solaris
libmqic.so.

If you want to use the programs on a machine that has only the WebSphere MQ client for Solaris installed, you must recompile the programs to link them with the client library:

```
$ /opt/SUNWspro/bin/cc -o <prog> <prog> c -mt -lmqic \
-lmqmcs -lsocket -lc -lnsl -ldl
```

The parameters must be entered in the correct order, as shown.

Windows
MQIC32.LIB.

Linking C++ applications with the WebSphere MQ client code

You can write applications to run on the client in C++. For information about how to link your C++ applications and for full details of all aspects of using C++, see WebSphere MQ Using C++.

Linking COBOL applications with the WebSphere MQ client code

AIX Link your COBOL application with the libmqicb.a library.

HP-UX
Link your COBOL application with the libmqicb.sl library.

If you are not using LU 6.2, consider linking to libsnastubs.a (in /opt/lib for HP-UX) to fully resolve function names. The need to link to this library depends on how you are using the -B flag during the linking stage. For more information see the WebSphere MQ Application Programming Guide.

Solaris
Link your COBOL application with the libmqicb.so library.

Windows
If you have a Windows COBOL application that you want to run in the client environment, link your application code with the MQICCBB library for 32-bit COBOL. The WebSphere MQ client for Windows does not support 16-bit COBOL.

Linking Visual Basic applications with the WebSphere MQ client code

You can link Visual Basic applications with the WebSphere MQ client code on Windows.

Link your Visual Basic application with the following include files:

CMQB.bas
MQI

CMQBB.bas
MQAI

CMQCFB.bas
PCF commands

CMQXB.bas
Channels

Set `mqtype=2` for the client in the Visual Basic compiler, to ensure the correct automatic selection of the client dll:

MQIC32.dll
Windows 2000, Windows XP and Windows 2003

Running applications on WebSphere MQ clients

This collection of topics explains the various ways in which an application running in a WebSphere MQ client environment can connect to a queue manager.

It covers the following topics:
- "Using environment variables" on page 130
- "Using the MQCNO structure" on page 130
- "Using DEFINE CHANNEL" on page 130
- "Role of the client channel definition table" on page 130
- "Examples of channel weighting and affinity" on page 135
- "Examples of MQCONN calls" on page 136

When an application running in a WebSphere MQ client environment issues an MQCONN or MQCONNX call, the client identifies how it is to make the connection. When an MQCONNX call is issued by an application on a WebSphere MQ client, the MQI client library searches for the client channel information in the following order:

1. Using the contents of the *ClientConnOffset* or *ClientConnPtr* fields of the MQCNO structure (if supplied). These identify the channel definition structure (MQCD) to be used as the definition of the client connection channel.
2. If the MQSERVER environment variable is set, the channel it defines is used.
3. If the MQCHLLIB and MQCHLTAB environment variables are set, the client channel definition table they point to is used.
4. Finally, if the environment variables are *not* set, the client searches for a client channel definition table whose path and name are established from the `DefaultPrefix` in the `mqs.ini` file or the Registry for Windows. If this fails, the client uses the following paths:
   - UNIX systems
     `/var/mqm/AMQCLCHL.TAB`

- Windows

  `C:\Program Files\IBM\Websphere MQ\amqclchl.tab`

The first of the options described above (using the *ClientConnOffset* or *ClientConnPtr* fields of MQCNO) is supported only by the MQCONNX call. If the application is using MQCONN rather than MQCONNX, the channel information is searched for in the remaining three ways in the order shown above. If the client fails to find any of these, the MQCONN or MQCONNX call fails.

The channel name (for the client connection) must match the server-connection channel name defined on the server for the MQCONN or MQCONNX call to succeed.

If you receive an MQRC_Q_MGR_NOT_AVAILABLE return code from your application with an error message in the error log file of AMQ9517 - File damaged, see "Migration and client channel definition tables" on page 95.

Using environment variables

Client channel information can be supplied to an application running in a client environment by the MQSERVER, MQCHLLIB, and MQCHLTAB environment variables.

See "MQSERVER" on page 114, "MQCHLLIB" on page 112 and "MQCHLTAB" on page 113 for details of these variables.

Using the MQCNO structure

You can specify the definition of the channel in a channel definition structure (MQCD), which is supplied using the MQCNO structure of the MQCONNX call.

For more information see "Using the MQCNO structure on an MQCONNX call" on page 92.

Using DEFINE CHANNEL

If you use the MQSC DEFINE CHANNEL command, the details you provide are placed in the client channel definition table. The contents of the *QMgrName* parameter of the MQCONN or MQCONNX call determines which queue manager the client connects to.

This file is accessed by the client to determine the channel an application will use. Where there is more than one suitable channel definition, the choice of channel is influenced by the client channel weight (CLNTWGHT) and connection affinity (AFFINITY) channel attributes.

Role of the client channel definition table

The client channel definition table is created when you define a queue manager.

Note: The same file can be used by more than one WebSphere MQ client. You access different versions of this file using the MQCHLLIB and MQCHLTAB WebSphere MQ environment variables. See "Using WebSphere MQ environment variables" on page 111 for information about environment variables.

Queue manager groups

When using a channel connection table, client applications have the flexibility to connect using a queue manager group name rather than a queue manager name. WebSphere MQ can then connect the client to any queue manager that is a member of the group.

You might choose to define connections to more than one server machine because:

- You want to connect a client to any one of a set of queue managers that is running, to improve availability.
- You want to reconnect a client to the same queue manager it connected to successfully last time, but connect to a different queue manager if the connection fails.
- You want to be able to retry a client connection to a different queue manager if the connection fails, by simply issuing the MQCONN in the client program again.
- You want to automatically reconnect a client connection to another queue manager if the connection fails, without writing any client code.
- You want to balance your client connections across a number of queue managers, with more clients connecting to some queue managers that others.
- You want to spread the reconnection of a large number of client connections over multiple queue managers and over a period of time, should a queue manager fail.
- You want to be able to move your queue managers without changing any client application code.
- You want to write client application programs that do not need to know queue manager names.

A queue manager group is a set of connections defined in the client channel connection table (CCDT). The set is defined by its members having the same value of the **QMNAME** attribute in their channel definitions.

Figure 27 on page 132 is a graphical representation of a client connection table, showing three queue manager groups, two named `QMNAME(`*`QM1`*`)` and `QMNAME(`*`QMGRP1`*`)`, and one blank, or default group.

1. Queue manager group *QM1* has three client connection channels, connecting it to QM1 and QM2.
2. The default queue manager group has six client connection channels connecting it to all the queue managers.
3. *QMGrp1* has client connection channels to two queue managers, QM4 and QM5.

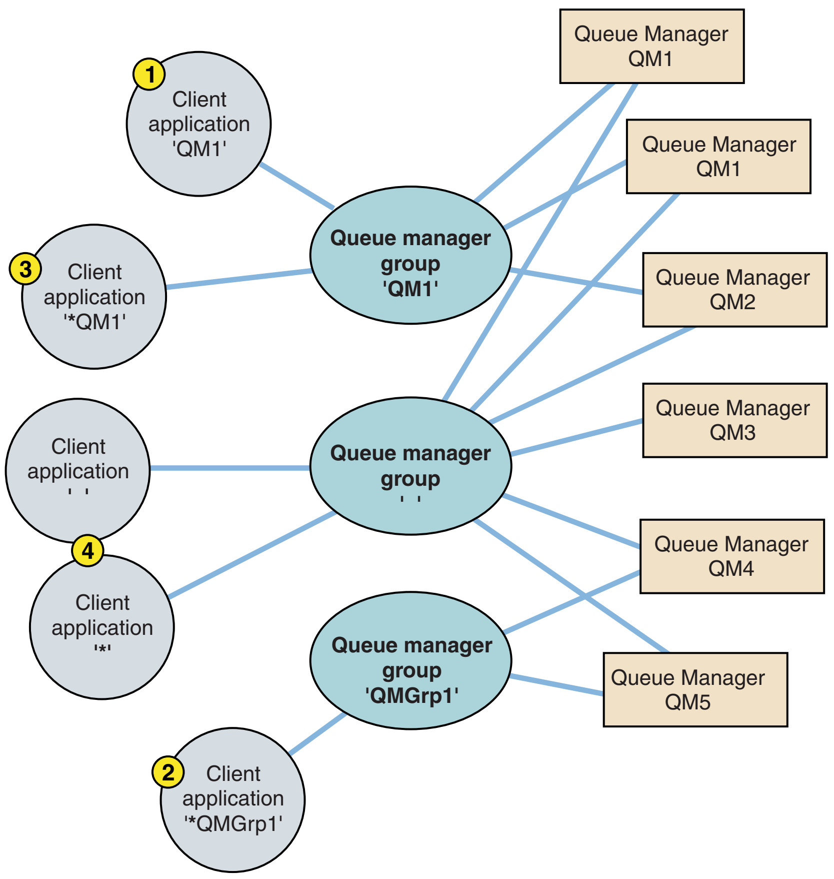

Figure 27. Queue manager groups

Four ways of using this client connection table are discussed below with the help of the numbered client applications in Figure 27.

1. In the first case, the client application passes a queue manager name, 'QM1', as the **QmgrName** parameter to its MQCONN or MQCONNX MQI call. The WebSphere MQ client code selects the matching queue manager group, QM1. The group contains three connection channels, and the WebSphere MQ client tries to connect to QM1 using each of these channels in turn looking for an WebSphere MQ listener for the connection attached to a running queue manager called QM1.

   The order of connection attempts depends on the value of the client connection AFFINITY attribute and the client channel weightings. Within these constraints, the order of connection attempts is randomized, both over the three possible connections, and over time, in order to spread out the load of making connections.

   The MQCONN or MQCONNX call issued by the client application succeeds when a connection is established to a running instance of QM1.
2. In the second case, the client application passes a queue manager name *prefixed with an asterisk*, '*QMGrp1' as the **QmgrName** parameter to its MQCONN or MQCONNX MQI call. The WebSphere MQ client selects the matching queue manager group, QMGrp1. This group contains two client connection channels, and the WebSphere MQ client tries to connect to *any* queue manager using each channel in turn. In this case the WebSphere MQ client is looking only for a successful connection - it does not matter what the name of the queue manager it connects to is.

The rule for the order of making connection attempts is the same as before. The only difference is that by prefixing the queue manager name by an asterisk, the client application is indicating that it does not care what the name of the queue manager it connects to is.

The MQCONN or MQCONNX call issued by the client application succeeds when a connection is established to a running instance of any queue manager connected to by the channels in the QMGrp1 queue manager group.

3. The third case is essentially the same as the second because the **QmgrName** parameter is prefixed by an asterisk, '*QM1'. The case illustrates that you can not determine which queue manager a client channel connection is going to connect to simply by inspecting the **QMNAME** attribute in one channel definition by itself. The fact that the **QMNAME** attribute of the channel definition is QM1, is not sufficient to require a connection is made to a queue manager called QM1. If your client application prefixes its **QmgrName** parameter with an asterisk then any queue manager is a possible connection target.

   In this case the MQCONN or MQCONNX calls issued by the client application succeed when a connection is established to a running instance of either QM1 or QM2.

4. The fourth example illustrates use of the default group. In this case the client application passes an asterisk, '*', or blank '   ', as the **QmgrName** parameter to its MQCONN or MQCONNX MQI call. By convention in the client channel definition, a blank **QMNAME** attribute signifies the default queue manager group and either a blank or asterisk **QmgrName** parameter matches a blank **QMNAME** attribute.

   In this example the default queue manager group has client channel connections to all the queue managers. By selecting the default queue manager group the application might be connected to any queue manager in the group.

   The MQCONN or MQCONNX call issued by the client application succeeds when a connection is established to a running instance of any queue manager.

   Note: The default group is completely different to a default queue manager, although an application uses a blank **QmgrName** parameter to connect to either the default queue manager or to the default queue manager. The concept of a default queue manager group is only relevant to a client application, and a default queue manager to a server application.

Define your client-connection channels on one queue manager only, including those channels that connect to a second or third queue manager. Do ***not*** define them on two queue managers and then try to merge the two client channel definition tables; this cannot be done. Only one client channel definition table can be accessed by the client.

Examples

Look at the list of reasons for using queue manager groups again at the beginning of the topic. How does using a queue manager group provide those capabilities?

Connect to any one of a set of queue managers.
Define a queue manager group with connections to all the queue managers in the set, and connect to the group using the **QmgrName** parameter prefixed by an asterisk.

Reconnect to the same queue manager, but connect to a different one, if the queue manager connected to last time is unavailable.
Define a queue manager group as before but set the attribute, **AFFINITY**(PREFERRED) on each client channel definition.

Retry a connection to a another queue manager if a connection fails.
Connect to a queue manager group, and reissue the MQCONN or MQCONNX MQI call if the connection is broken or the queue manager fails.

Automatically reconnect to another queue manager if a connection fails.
Connect to a queue manager group using the MQCONNX **MQCNO** option MQCNO_RECONNECT, or set the environment variable **MQ_AUTO_RECONNECT** to reconnect a client that uses the standard MQCONN MQI call.

Balance client connections across queue managers, with more clients connected to some queue managers than others.
Define a queue manager group, and set the **CLNTWGHT** attribute on each client channel definition to distribute the connections unevenly.

Spread the client reconnection load unevenly, and spread over a period of time, after a connection or queue manager failure.
Do the same as the preceding example. The WebSphere MQ client randomizes reconnections across queue managers and spreads the reconnections over time.

Move your queue managers without changing any client code.
The CCDT isolates your client application from the location of the queue manager.

You have a choice of either distributing the client connection table to each client, or placing the CCDT on a shared file system for each client to refer to, or using the programmatic version of the CCDT supported in the MQCONNX MQI call, and calling a service to pass the CCDT to the client application.

Write a client application that does not know queue manager names.
Use queue manager group names and establish a naming convention for queue manager group names that is relevant to your client applications in your organization, and reflects the architecture of your solutions rather than the naming of queue managers.

Queue-sharing groups

You can connect your application to a queue manager that is part of a queue-sharing group. This can be done by using the queue-sharing group name instead of the queue manager name on the MQCONN or MQCONNX call.

Queue-sharing groups have a name of up to four characters. The name must be unique in your network, and must be different from any queue manager names.

The client channel definition should use the queue sharing group generic interface to connect to an available queue manager in the group. For more information, see "Connecting to the generic interface" on page 99. A check is made to ensure that the queue manager the listener connects to is a member of the queue sharing group.

For more information on shared queues, see the see the WebSphere MQ for z/OS Concepts and Planning Guide book, and the WebSphere MQ Intercommunication book.

Examples of channel weighting and affinity

The ClientChannelWeight and ConnectionAffinity channel attributes control how client-connection channels are selected when more than one suitable channel is available for a connection. These channels are configured to connect to different queue managers in order to provide higher availability, workload balancing, or both. Note that MQCONN calls which could result in a connection to one of several queue managers, must prefix the queue manager name with an asterisk as described in:Examples of MQCONN calls: Example 1. Queue manager name includes an asterisk (*).

Applicable candidate channels for a connection are those whose QMNAME attribute matches the queue manager name specified in the MQCONN call. If all applicable channels for a connection have a ClientChannelWeight of zero (the default) then they are selected in alphabetical order as in the example: Examples of MQCONN calls: Example 1. Queue manager name includes an asterisk (*).

The following examples illustrate what happens when non-zero ClientChannelWeights are used. Note that, since this feature involves pseudo-random channel selection, the examples show a sequence of actions that might happen rather than what definitely will.

Example 1. The ConnectionAffinity attribute is set to PREFERRED

In this example a number of client machines use a Client Channel Definition Table (CCDT) provided by a queue manager. The CCDT includes the following CLNTCONN channels:

```
.
.
CHANNEL(A) QMNAME(DEV) CONNAME(devqm.it.company.example)
CHANNEL(B) QMNAME(CORE) CONNAME(core1.ops.company.example) CLNTWGHT(5) +
AFFINITY(PREFERRED)
CHANNEL(C) QMNAME(CORE) CONNAME(core2.ops.company.example) CLNTWGHT(3) +
AFFINITY(PREFERRED)
CHANNEL(D) QMNAME(CORE) CONNAME(core3.ops.company.example) CLNTWGHT(2) +
AFFINITY(PREFERRED).
.
.
```

The application issues MQCONN(*CORE)

Channel A is not a candidate for this connection, because the QMNAME attribute does not match. Channels B, C and D are identified as candidates, and are placed in an order of preference based on their weighting. In this example the order might be C, B, D. The client will attempt to connect to the queue manager at core2.ops.company.example. The name of the queue manager at that address is not checked, because the MQCONN call included an asterisk in the queue manager name.

It is important to note that, with `AFFINITY(PREFERRED)`, each time this particular client machine connects it will place the channels in the same initial order of preference. This applies even when the connections are from different processes or at different times.

In this example, the queue manager at core.2.ops.company.example cannot be reached. The client will attempt to connect to core1.ops.company.example because channel B is next in the order of preference. In addition, channel C will be demoted to become the least preferred.

A second MQCONN(*CORE) call is issued by the same application. Channel C was demoted by the previous connection, so the most preferred channel is now B. This connection will be made to core1.ops.company.example.

A second machine sharing the same Client Channel Definition Table may place the channels in a different initial order of preference. For example, D, B, C. Under normal circumstances, with all channels working, applications on this machine will be connected to core3.ops.company.example while those on the machine described above will be connected to core2.ops.company.example. This allows workload balancing of large numbers of clients across multiple queue managers while allowing each individual client to connect to the same queue manager if it is available.

Example 2. The ConnectionAffinity attribute is set to NONE

In this example a number of client machines use a Client Channel Definition Table (CCDT) provided by a queue manager. The CCDT includes the following CLNTCONN channels:

```
.
.
CHANNEL(A) QMNAME(DEV) CONNAME(devqm.it.company.example)
CHANNEL(B) QMNAME(CORE) CONNAME(core1.ops.company.example) CLNTWGHT(5) +
AFFINITY(NONE)
CHANNEL(C) QMNAME(CORE) CONNAME(core2.ops.company.example) CLNTWGHT(3) +
AFFINITY(NONE)
CHANNEL(D) QMNAME(CORE) CONNAME(core3.ops.company.example) CLNTWGHT(2) +
AFFINITY(NONE).
.
.
```

The application ussues MQCONN(*CORE). As in the previous example, channel A is not considered because the QMNAME does not match. Either channel B, C or D are selected based on their weighting, with a probability of 50%, 30% or 20% respectively. In this example, B might be selected. There is no persistent order of preference created.

A second MQCONN(*CORE) call is made. Again, one of the three applicable channels is selected, with the same probabilities. In this example, C is chosen. However, core2.ops.company.example does not respond, an so another choice is made between the remaining candidate channels. B is selected and the application is connected to core1.ops.company.example.

With AFFINITY(NONE), each MQCONN call is independant of any other. Therefore when this example application makes a third MQCONN(*CORE), it might once more attempt to connect through the broken channel C, before choosing one of B or D.

Examples of MQCONN calls

In each of the following examples, the network is the same; there is a connection defined to two servers from the same WebSphere MQ client. (In these examples, the MQCONNX call could be used instead of the MQCONN call.)

There are two queue managers running on the server machines, one named SALE and the other named SALE_BACKUP.

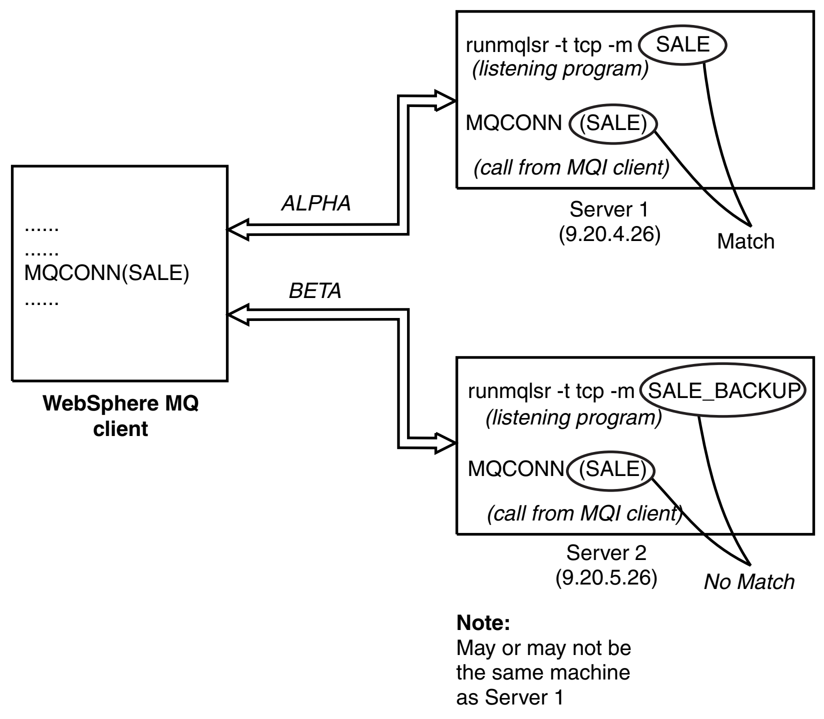

Figure 28. MQCONN example

The definitions for the channels in these examples are:

SALE definitions:

```
DEFINE CHANNEL(ALPHA) CHLTYPE(SVRCONN) TRPTYPE(TCP) +
DESCR('Server connection to WebSphere MQ client')

DEFINE CHANNEL(APLHA) CHLTYPE(CLNTCONN) TRPTYPE(TCP) +
CONNAME(9.20.4.26) DESCR('WebSphere MQ client connection to server 1') +
QMNAME(SALE)

DEFINE CHANNEL(BETA) CHLTYPE(CLNTCONN) TRPTYPE(TCP) +
CONNAME(9.20.5.26) DESCR('WebSphere MQ client connection to server 2') +
QMNAME(SALE)
```

SALE_BACKUP definition:

```
DEFINE CHANNEL(BETA) CHLTYPE(SVRCONN) TRPTYPE(TCP) +
DESCR('Server connection to WebSphere MQ client')
```

The client channel definitions can be summarized as follows:

Name	CHLTYPE	TRPTYPE	CONNAME	QMNAME
ALPHA	CLNTCONN	TCP	9.20.4.26	SALE
BETA	CLNTCONN	TCP	9.20.5.26	SALE

What the examples demonstrate

Suppose the communication link to Server 1 is temporarily broken. The use of multiple queue managers as a backup system is demonstrated.

Each example covers a different MQCONN call and gives an explanation of what happens in the specific example presented, by applying the following rules:

1. WebSphere MQ searches the client channel definition table, in alphabetic channel name order, looking in the queue manager name (QMNAME) field for an entry corresponding to the one given in the MQCONN call.
2. If a match is found, the channel definition is used.
3. An attempt is made to start the channel to the machine identified by the connection name (CONNAME). If this is successful, the application continues. It requires:
   - A listener to be running on the server.
   - The listener to be connected to the same queue manager as the one the client wishes to connect to (if specified).
4. If the attempt to start the channel fails and there is more than one entry in the client channel definition table (in this example there are two entries), the file is searched for a further match. If a match is found, processing continues at step 1.
5. If no match is found, or there are no more entries in the client channel definition table and the channel has failed to start, the application is unable to connect. An appropriate reason code and completion code are returned in the MQCONN call. The application can take action based on the reason and completion codes returned.

Example 1. Queue manager name includes an asterisk (*)

In this example the application is not concerned about which queue manager it connects to. The application issues an MQCONN call for a queue manager name including an asterisk. A suitable channel is chosen.

The application issues:

```
MQCONN (*SALE)
```

Following the rules, this is what happens in this instance:

1. The client channel definition table is scanned for the queue manager name SALE, matching with the application MQCONN call.
2. Channel definitions for ALPHA and BETA are found.
3. If one channel has a CLNTWGHT value of 0, this channel is selected. If both have a CLNTWGHT value of 0, channel ALPHA is selected because it is first in alphabetic sequence. If both channels have a non-zero CLNTWGHT value, one channel is randomly selected, based on its weighting.
4. An attempt to start the channel is made.
5. If channel BETA was selected, the attempt to start it is successful.
6. If channel ALPHA was selected, the attempt to start it is NOT successful because the communication link is broken. The following steps then apply:
   a. The only other channel for the queue manager name SALE is BETA.
   b. An attempt to start this channel is made – this is successful.
7. A check to see that a listener is running shows that there is one running. It is not connected to the SALE queue manager, but because the MQI call parameter

has an asterisk (*) included in it, no check is made. The application is connected to the `SALE_BACKUP` queue manager and continues processing.

Example 2. Queue manager name specified

The application requires a connection to a specific queue manager, named `SALE`, as seen in the MQI call:

```
MQCONN (SALE)
```

Following the rules, this is what happens in this instance:

1. The client channel definition table is scanned in alphabetic channel name sequence, for the queue manager name `SALE`, matching with the application MQCONN call.
2. The first channel definition found to match is `ALPHA`.
3. An attempt to start the channel is made – this is *not* successful because the communication link is broken.
4. The client channel definition table is again scanned for the queue manager name `SALE` and the channel name `BETA` is found.
5. An attempt to start the channel is made – this is successful.
6. A check to see that a listener is running shows that there is one running, but it is not connected to the `SALE` queue manager.
7. There are no further entries in the client channel definition table. The application cannot continue and receives return code MQRC_Q_MGR_NOT_AVAILABLE.

Example 3. Queue manager name is blank or an asterisk (*)

In this example the application is not concerned about which queue manager it connects to. This is treated in the same way as "Example 1. Queue manager name includes an asterisk (*)" on page 138.

Note: If this application were running in an environment other than a WebSphere MQ client, and the name was blank, it would be attempting to connect to the default queue manager. This is *not* the case when it is run from a client environment; the queue manager accessed is the one associated with the listener to which the channel connects.

The application issues:

```
MQCONN ("")
```

or

```
MQCONN (*)
```

Following the rules, this is what happens in this instance:

1. The client channel definition table is scanned in alphabetic channel name sequence, for a queue manager name that is blank, matching with the application MQCONN call.
2. The entry for the channel name `ALPHA` has a queue manager name in the definition of `SALE`. This does *not* match the MQCONN call parameter, which requires the queue manager name to be blank.
3. The next entry is for the channel name `BETA`.

4. The `queue manager name` in the definition is `SALE`. Once again, this does *not* match the MQCONN call parameter, which requires the queue manager name to be blank.
5. There are no further entries in the client channel definition table. The application cannot continue and receives return code MQRC_Q_MGR_NOT_AVAILABLE.

Preparing and running extended transactional client applications

This chapter contains the following sections:
- "Preparing and running CICS and Tuxedo applications"
- "Preparing and running Microsoft Transaction Server applications" on page 142
- "Preparing and running WebSphere MQ JMS applications" on page 142

Preparing and running CICS and Tuxedo applications

To prepare CICS and Tuxedo applications to run as WebSphere MQ client applications, follow the instructions in the WebSphere MQ Application Programming Guide.

Note, however, that the information in the WebSphere MQ Application Programming Guide that deals specifically with preparing CICS and Tuxedo applications, including the sample programs supplied with WebSphere MQ, assumes that you are preparing applications to run on a WebSphere MQ server system. As a result, the information refers only to WebSphere MQ libraries that are intended for use on a server system. When you are preparing your client applications, you must do the following therefore:
- Use the appropriate client system library for the language bindings that your application uses. For example, for applications written in C on AIX, HP-UX, or Solaris, use the library libmqic instead of libmqm and, on Windows systems, use the library mqic.lib instead of mqm.lib.
- Instead of the server system libraries shown in Table 20, for AIX, HP-UX, and Solaris, and Table 21, for Windows systems, use the equivalent client system libraries. If a server system library is not listed in these tables, use the same library on a client system.

Table 20. Client system libraries on AIX, HP-UX, and Solaris

Library for a WebSphere MQ server system	Equivalent library to use on a WebSphere MQ client system
libmqmxa	libmqcxa

Table 21. Client system libraries on Windows systems

Library for a WebSphere MQ server system	Equivalent library to use on a WebSphere MQ client system
mqmxa.lib	mqcxa.lib
mqmtux.lib	mqcxa.lib
mqmenc.lib	mqcxa.lib
mqmcics4.lib	mqccics4.lib

Sample programs

Table 22 lists the CICS and Tuxedo sample programs that are supplied for use on AIX, HP-UX, and Solaris client systems. Table 23 lists the equivalent information for Windows client systems. The tables also list the files that are used for preparing and running the programs. For a description of the sample programs, see the WebSphere MQ Application Programming Guide.

Table 22. Sample programs for AIX, HP-UX, and Solaris client systems

Description	Source	Executable module
CICS program	amqscic0.ccs	amqscicc
Header file for the CICS program	amqscih0.h	-
Tuxedo client program to put messages	amqstxpx.c	-
Tuxedo client program to get messages	amqstxgx.c	-
Tuxedo server program for the two client programs	amqstxsx.c	-
UBBCONFIG file for the Tuxedo programs	ubbstxcx.cfg	-
Field table file for the Tuxedo programs	amqstxvx.flds	-
View description file for the Tuxedo programs	amqstxvx.v	-

Table 23. Sample programs for Windows client systems

Description	Source	Executable module
CICS transaction	amqscic0.ccs	amqscicc
Header file for the CICS transaction	amqscih0.h	-
Tuxedo client program to put messages	amqstxpx.c	-
Tuxedo client program to get messages	amqstxgx.c	-
Tuxedo server program for the two client programs	amqstxsx.c	-
UBBCONFIG file for the Tuxedo programs	ubbstxcx.cfg	-
Field table file for the Tuxedo programs	amqstxvx.fld	-
View description file for the Tuxedo programs	amqstxvx.v	-
Makefile for the Tuxedo programs	amqstxmc.mak	-
ENVFILE file for the Tuxedo programs	amqstxen.env	-

Error log messages

When you run CICS or Tuxedo applications that use an extended transactional client, the messages that you might see in the WebSphere MQ error log files are documented in WebSphere MQ Messages. One of the messages, AMQ5203, has been modified for use with an extended transactional client. Here is the text of the modified message:

AMQ5203: An error occurred calling the XA interface.:

Explanation

The error number is &2 where a value of 1 indicates the supplied flags value of &1 was invalid, 2 indicates that there was an attempt to use threaded and non-threaded libraries in the same process, 3 indicates that there was an error with the supplied queue manager name ’&3’, 4 indicates that the resource manager id of &1 was invalid, 5 indicates that an attempt was made to use a second queue manager called ’&3’ when another queue manager was already connected, 6 indicates that the Transaction Manager has been called when the application isn’t connected to a queue manager, 7 indicates that the XA call was made while another call was in progress, 8 indicates that the xa_info string ’&4’ in the xa_open call contained an invalid parameter value for parameter name ’&5’, and 9 indicates that the xa_info string ’&4’ in the xa_open call is missing a required parameter, parameter name ’&5’.

User response

Correct the error and try the operation again.

Preparing and running Microsoft Transaction Server applications

For general information about how to develop Microsoft Transaction Server (MTS) applications that access WebSphere MQ resources, see the section on MTS in the WebSphere MQ Help Center.

To prepare an MTS application to run as a WebSphere MQ client application, do one of the following for each component of the application:

- If the component uses the C language bindings for the MQI, follow the instructions in the WebSphere MQ Application Programming Guide but link the component with the library mqicxa.lib instead of mqic.lib.
- If the component uses the WebSphere MQ C++ classes, follow the instructions in WebSphere MQ Using C++ but link the component with the library imqx23vn.lib instead of imqc23vn.lib.
- If the component uses the Visual Basic language bindings for the MQI, follow the instructions in the WebSphere MQ Application Programming Guide but, when you define the Visual Basic project, type `MqType=3` in the **Conditional Compilation Arguments** field.
- If the component uses the WebSphere MQ Automation Classes for ActiveX (MQAX), define an environment variable, GMQ_MQ_LIB, with the value `mqic32xa.dll` .

  You can define the environment variable from within your application, or you can define it so that its scope is system wide. However, defining it as system wide can cause any existing MQAX application, that does not define the environment variable from within the application, to behave incorrectly.

Preparing and running WebSphere MQ JMS applications

To prepare and run WebSphere MQ JMS applications in client mode, with WebSphere Application Server as your transaction manager, follow the instructions in WebSphere MQ Using Java.

When you run a WebSphere MQ JMS client application, you might see the following warning messages:

MQJE080
Insufficient license units - run setmqcap

MQJE081
File containing the license unit information is in the wrong format - run setmqcap

MQJE082
File containing the license unit information could not be found - run setmqcap

Solving problems

This chapter discusses the following topics:
- "WebSphere MQ client fails to make a connection"
- "Stopping WebSphere MQ clients" on page 144
- "Error messages with WebSphere MQ clients" on page 144
- "Using trace on Windows" on page 144
- "Using trace on UNIX systems" on page 145

An application running in the WebSphere MQ client environment receives MQRC_* reason codes in the same way as WebSphere MQ server applications. However, there are additional reason codes for error conditions associated with WebSphere MQ clients. For example:
- Remote machine not responding
- Communications line error
- Invalid machine address

The most common time for errors to occur is when an application issues an MQCONN or MQCONNX and receives the response MQRC_Q_MQR_NOT_AVAILABLE. Look in the client error log for a message explaining the failure. There might also be errors logged at the server, depending on the nature of the failure. Also, check that the application on the WebSphere MQ client is linked with the correct library file.

WebSphere MQ client fails to make a connection

When the WebSphere MQ client issues an MQCONN or MQCONNX call to a server, socket and port information is exchanged between the WebSphere MQ client and the server. For any exchange of information to take place, there must be a program on the server machine whose role is to 'listen' on the communications line for any activity. If there is no program doing this, or there is one but it is not configured correctly, the MQCONN or MQCONNX call fails, and the relevant reason code is returned to the WebSphere MQ client application.

If the connection is successful, WebSphere MQ protocol messages are exchanged and further checking takes place. During the WebSphere MQ protocol checking phase, some aspects are negotiated while others cause the connection to fail. It is not until all these checks are successful that the MQCONN or MQCONNX call succeeds.

For information about the MQRC_* reason codes, see the WebSphere MQ Application Programming Reference manual.

Stopping WebSphere MQ clients

Even though a WebSphere MQ client has stopped, it is still possible for the associated process at the server to be holding its queues open. The queues will be closed when the communications layer detects that the partner has gone.

If sharing conversations is enabled, the server channel is always in the correct state for the communications layer to detect that the partner has gone.

Error messages with WebSphere MQ clients

When an error occurs with a WebSphere MQ client system, error messages are put into the WebSphere MQ system error files.

- On UNIX these files are found in the /var/mqm/errors directory
- On Windows, these files are found in the errors subdirectory of the WebSphere MQ client installation. Typically this directory is C:\Program Files\IBM\WebSphere MQ\errors

Certain client errors can also be recorded in the WebSphere MQ error files associated with the server to which the client was connected.

Using trace on Windows

Use the strmqtrc and endmqtrc commands to start and end tracing.

A client on Windows uses the following commands for the client trace facility:

strmqtrc
to start tracing

endmqtrc
to end tracing

For more information about the trace commands, see Tracing in the *WebSphere MQ System Administration Guide*.

The output files are created in the *mqmtop*\trace directory.

File names for trace files

Trace file names are constructed in the following way:

```
AMQppppp.qq.TRC
```

where ppppp is the process ID (PID) of the process producing the trace and qq is a sequence number. The value of qq starts at 0. If the full filename already exists, this value is incremented by one until a unique trace filename is found. A trace filename can already exist if a process is reused.

Note:

1. The value of the process ID can contain fewer or more digits than shown in the example.
2. There is one trace file for each process running as part of the entity being traced.

SSL trace files have the names AMQ.SSL.TRC and AMQ.SSL.TRC.1. You cannot format SSL trace files; send them unchanged to IBM support.

How to examine First Failure Support Technology (FFST) files

The files are produced already formatted and are in the errors subdirectory of the WebSphere MQ client installation directory.

These are normally severe, unrecoverable errors and indicate either a configuration problem with the system or a WebSphere MQ internal error.

The files are named `AMQnnnnn.mm.FDC`, where:
- `nnnnn` is the process id reporting the error
- `mm` is a sequence number, normally 0

When a process creates an FFST™ it also sends a record to the system log. The record contains the name of the FFST file to assist in automatic problem tracking.

The system log entry is made at the "user.error" level.

First Failure Support Technology™ is explained in detail in the WebSphere MQ System Administration Guide manual.

Using trace on UNIX systems

UNIX systems use the following commands for the WebSphere MQ client trace facility.

strmqtrc
to start tracing

endmqtrc
to end tracing

dspmqtrc <filename>
to display a formatted trace file

| For more information about the trace commands, see Tracing in the *WebSphere MQ*
| *System Administration Guide*.

The trace facility uses a number of files, which are:
- One file for each entity being traced, in which trace information is recorded
- One additional file on each machine, to provide a reference for the shared memory used to start and end tracing
- One file to identify the semaphore used when updating the shared memory

Files associated with trace are created in a fixed location in the file tree, which is `/var/mqm/trace`.

All client tracing takes place to files in this directory.

You can handle large trace files by mounting a temporary file system over this directory.

File names for trace files

Trace file names are constructed in the following way:

```
AMQppppp.qq.TRC
```

where ppppp is the process ID (PID) of the process producing the trace and qq is a sequence number. The value of qq starts at 0. If the full filename already exists, this value is incremented by one until a unique trace filename is found. A trace filename can already exist if a process is reused..

Note:

1. The value of the process ID can contain fewer or more digits than shown in the example.
2. There is one trace file for each process running as part of the entity being traced.

SSL trace files have the names AMQ.SSL.TRC and AMQ.SSL.TRC.1. You cannot format SSL trace files; send them unchanged to IBM support.

How to examine First Failure Support Technology (FFST) files

FFST logs are written when a severe WebSphere MQ error occurs. They are written to the directory `/var/mqm/errors`.

These are normally severe, unrecoverable errors and indicate either a configuration problem with the system or a WebSphere MQ internal error.

The files are named `AMQnnnnn.mm.FDC`, where:

- `nnnnn` is the process id reporting the error
- `mm` is a sequence number, normally 0

When a process creates an FFST it also sends a record to the system log. The record contains the name of the FFST file to assist in automatic problem tracking.

The system log entry is made at the "user.error" level.

First Failure Support Technology is explained in detail in the WebSphere MQ System Administration Guide.

Using trace on AIX

As well as the method described in "Using trace on UNIX systems" on page 145, WebSphere MQ for AIX can use the standard AIX system trace.

This is explained in detail in the WebSphere MQ System Administration Guide.

Chapter 5. A review of transaction management

A *resource manager* is a computer subsystem that owns and manages resources that can be accessed and updated by applications. The following are examples of resource managers:

- A WebSphere MQ queue manager, whose resources are its queues
- A DB2 database, whose resources are its tables

When an application updates the resources of one or more resource managers, there might be a business requirement to ensure that certain updates all complete successfully as a group, or none of them complete. The reason for this kind of requirement is that the business data would be left in an inconsistent state if some of these updates completed successfully, but others did not.

Updates to resources that are managed in this way are said to occur within a *unit of work*, or a *transaction*. During a unit of work, an application issues requests to resource managers to update their resources. The unit of work ends when the application issues a request to commit all the updates. Until the updates are committed, none of them become visible to other applications that are accessing the same resources. Alternatively, if the application decides that it cannot complete the unit of work for any reason, it can issue a request to back out all the updates it has requested up to that point. In this case, none of the updates ever become visible to other applications.

The point in time when all the updates within a unit of work are either committed or backed out is called a *syncpoint*. An update within a unit of work is said to occur *within syncpoint control*. If an application requests an update that is *outside of syncpoint control*, the resource manager commits the update immediately, even if there is a unit of work in progress, and the update cannot be backed out subsequently.

The computer subsystem that manages units of work is called a *transaction manager*, or a *syncpoint coordinator*. A transaction manager is responsible for ensuring that all updates to resources within a unit of work complete successfully, or none of them complete. It is to a transaction manager that an application issues a request to commit or back out a unit of work. Examples of transaction managers are CICS and WebSphere Application Server, although both of these possess other function as well.

Some resource managers provide their own transaction management function. For example, a WebSphere MQ queue manager can manage units of work involving updates to its own resources and updates to DB2 tables. The queue manager does not need a separate transaction manager to perform this function, although one can be used if it is a user requirement. If a separate transaction manager is used, it is referred to as an *external transaction manager*.

For an external transaction manager to manage a unit of work, there must be an architected interface between the transaction manager and every resource manager that is participating in the unit of work. This interface allows the transaction manager and a resource manager to communicate with each other. One of these interfaces is the *XA Interface*, which is a standard interface supported by a number of transaction managers and resource managers. The XA Interface is published by The Open Group in *Distributed Transaction Processing: The XA Specification*.

When more than one resource manager participates in a unit of work, a transaction manager must use a *two phase commit* protocol to ensure that all the updates within the unit of work complete successfully or none of them complete, even if there is a system failure. When an application issues a request to a transaction manager to commit a unit of work, the transaction manager does the following:

Phase 1 (Prepare to commit)
The transaction manager asks each resource manager participating in the unit of work to ensure that all the information about the intended updates to its resources is in a recoverable state. A resource manager normally does this by writing the information to a log and ensuring that the information is written through to hard disk. Phase 1 completes when the transaction manager receives notification from each resource manager that the information about the intended updates to its resources is in a recoverable state.

Phase 2 (Commit)
When Phase 1 is complete, the transaction manager makes the irrevocable decision to commit the unit of work. It asks each resource manager participating in the unit of work to commit the updates to its resources. When a resource manager receives this request, it must commit the updates. It does not have the option to back them out at this stage. Phase 2 completes when the transaction manager receives notification from each resource manager that it has committed the updates to its resources.

The XA Interface uses a two phase commit protocol.

Notices

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.

GC34-6934-01

Spine information:

IBM WebSphere MQ Clients Version 7.0