WebSphere MQ

Clients

GC34-6590-00

WebSphere MQ

Clients

GC34-6590-00

Note!
Before using this information and the product it supports, be sure to read the general information under Appendix B, "Notices," on page 145.

First edition (May 2005)

Note
This is a draft edition of ″Clients″ for WebSphere MQ Version 6.0. A final edition will be published as soon as possible. If you have any comments on this book, please contact IBM using any of the methods listed in the appendix **Sending your comments to IBM**.

This edition of the book applies to the WebSphere MQ Version 6.0 clients supplied with the following product:

- WebSphere MQ Version 6.0
- WebSphere MQ for z/OS Version 6.0

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

Contents

Figures vii

Tables ix

About this book xi
What you need to know to understand this book . . xi
Terms used in this book xi
How to use this book xii

Summary of changes xiii
Changes for this edition (GC34–6590–00) xiii

Part 1. Introduction to WebSphere MQ clients 1

Chapter 1. Overview of WebSphere MQ clients 3
What is a WebSphere MQ client? 3
How the client connects to the server 4
What is an extended transactional client? 5
Why use WebSphere MQ clients? 6
What applications run on a WebSphere MQ client? 6
How do I set up a WebSphere MQ client? 6

Part 2. Installing WebSphere MQ clients 9

Chapter 2. Preparing for installation . . 11
Platform support for WebSphere MQ clients . . . 11
Platform support for extended transactional clients 11
Applications on WebSphere MQ Version 6.0 clients 13
Communications 13
Performance considerations 13
Hardware and software requirements. 14
AIX client: hardware and software required . . 15
HP-UX client: hardware and software required 16
Linux client: hardware and software required . . 17
Solaris client: hardware and software required. . 19
Windows client: hardware and software required 20

Chapter 3. Installing client components 21
Installing a WebSphere MQ client and server system 21
Installing from an electronic software download 22
Installing WebSphere MQ clients on the same machine as the server 22
Uninstalling WebSphere MQ clients 23
Installing on AIX 24
Components for AIX 24
Before installing 24
Custom installation. 26
Installing without SSL support 27
Installing the extended transactional function . . 27
Migrating from an earlier version of WebSphere MQ for AIX 28
Changing the national language 29
Translated messages 30
Removing a WebSphere MQ client from AIX . . 30
Migrating to and from the WebSphere MQ SSL support. 30
Uninstalling the extended transactional function 30
Installing on HP-UX 32
Components for HP-UX 32
Before installation 32
Installation 33
Installing the extended transactional function . . 33
Translated messages 34
Removing a WebSphere MQ client from HP-UX 34
Migrating to and from the WebSphere MQ SSL support. 34
Uninstalling the extended transactional function 34
Installing on Linux 36
Components for Linux. 36
Before installation 37
Installation 38
Installing the extended transactional function . . 39
Translated messages 40
Removing the WebSphere MQ client from Linux 40
Migrating to and from the WebSphere MQ SSL support. 41
Uninstalling the extended transactional function 41
Installing on Solaris 42
Components for Solaris 42
Before installation 42
Installation 43
Installing the extended transactional function . . 43
Translated messages 44
Removing a WebSphere MQ client from Solaris 45
Migrating to and from the WebSphere MQ SSL support. 45
Uninstalling the extended transactional function 45
Installing on Windows. 46
Preparing to install the WebSphere MQ client . . 46
Installing the WebSphere MQ client 46
Other methods of installing the WebSphere MQ client 50
Uninstalling a WebSphere MQ client 61

Chapter 4. Configuring communication links. 67
Deciding which communication type to use . . . 67
Defining a TCP/IP connection 69
Defining a TCP/IP connection on a WebSphere MQ client 69
Defining a TCP/IP connection on a WebSphere MQ server. 69

TCP/IP connection limits 69
Defining an LU 6.2 connection 70
Defining an LU 6.2 connection on a WebSphere MQ client 70
Defining an LU 6.2 Connection on a WebSphere MQ server 70
Defining a NetBIOS connection 70
Defining an SPX connection 70

Chapter 5. Configuring an extended transactional client 71
XA compliant transaction managers 71
The xa_open string 73
The XA switch structures 76
Configuring for CICS 77
Configuring for Encina 78
Configuring for Tuxedo 79
Microsoft Transaction Server 79
WebSphere Application Server 80

Chapter 6. Verifying the installation . . 83
The installation used for the example 83
What the example shows 83
Setting up the server on UNIX and Windows systems 83
Setting up a Windows server using WebSphere MQ Explorer 84
Setting up the server on i5/OS 85
Setting up the server on z/OS 85
Setting up the WebSphere MQ client 86
Defining a client-connection channel using MQSERVER 86
Defining a client-connection channel using WebSphere MQ Explorer 86
Putting a message on the queue 87
On the WebSphere MQ client workstation . . . 87
Getting the message from the queue 87
On the WebSphere MQ client workstation . . . 87
Ending verification 87

Part 3. System administration . . . 89

Chapter 7. Setting up WebSphere MQ client security. 91
Authentication 91
User IDs 92
Access control 92

Chapter 8. Using channels 95
What is a channel? 95
Message Channels 95
MQI Channels 95
Defining MQI channels 96
Automatically defined channels 96
User defined channels 97
Creating one definition on the WebSphere MQ client and the other on the server 97
On the server 97
On the WebSphere MQ client 98
Creating both definitions on the server 99
Defining the server-connection channel 100
Defining the client-connection channel 100
Accessing client-connection channel definitions 101
Client channel definition table 101
Migrating to a later release level of WebSphere MQ. 102
Channel exits 103
Path to exits 103
Connecting a client to a queue-sharing group . . 103
Connecting to a specific queue manager . . . 104
Connecting to the generic interface 104
Stopping channels 104

Chapter 9. The Secure Sockets Layer (SSL) on WebSphere MQ clients . . . 107
Specifying that an MQI channel uses SSL 107
Specifying the location of LDAP servers that hold certificate revocation lists (CRLs) 108
When a WebSphere MQ client application issues an MQCONNX call 108
Using a client channel definition table 108
Using Active Directory on Windows. 109
Renegotiating the secret key 109
Refreshing a client's view of the SSL key repository contents and SSL settings 110
Specifying that only FIPS-certified cryptography will be used 110

Chapter 10. Using WebSphere MQ environment variables. 111
MQCCSID 112
MQCHLLIB 112
MQCHLTAB. 113
Using MQCHLLIB and MQCHLTAB 113
MQIPADDRV 113
MQNAME 113
MQSERVER 114
TCP/IP default port 114
SPX default socket. 114
Using MQSERVER. 115
Canceling MQSERVER 115
MQSSLCRYP 115
MQSSLFIPS 116
MQSSLKEYR 116
MQSSLRESET 116

Part 4. Application programming 117

Chapter 11. Using the message queue interface (MQI) 119
Limiting the size of a message. 119
Choosing client or server coded character set identifier (CCSID) 119
CCSID and encoding fields - multiple puts . . 120
Designing applications 120
Using MQINQ 120
Using syncpoint coordination 120
Using MQCONNX 121

Shared connection handles on MQCONNX . . 121

Chapter 12. Building applications for WebSphere MQ clients 123
Running applications in the WebSphere MQ client environment. 123
Triggering in the client environment. 124
Process definition 124
Trigger monitor 124
CICS applications (non-z/OS) 125
Linking C applications with the WebSphere MQ client code 125
Linking C++ applications with the WebSphere MQ client code 126
Linking COBOL applications with the WebSphere MQ client code 126
Linking Visual Basic applications with the WebSphere MQ client code 126

Chapter 13. Running applications on WebSphere MQ clients 129
Using environment variables 129
Using the MQCNO structure 130
Using DEFINE CHANNEL 130
Role of the client channel definition table 130
Multiple queue managers 130
Queue-sharing groups 130
Examples of MQCONN calls 131
What the examples demonstrate 132
Example 1. Queue manager name includes an asterisk (*) 132
Example 2. Queue manager name specified . . 133
Example 3. Queue manager name is blank or an asterisk (*) 133

Chapter 14. Preparing and running extended transactional client applications 135
Preparing and running CICS, Encina, and Tuxedo applications 135
Sample programs 135
Error log messages 137
Preparing and running Microsoft Transaction Server applications 137
Preparing and running WebSphere MQ JMS applications 137

Chapter 15. Solving problems 139
WebSphere MQ client fails to make a connection 139
Stopping WebSphere MQ clients 139
Error messages with WebSphere MQ clients . . . 140
Using trace on Windows 140
File names for trace files. 140
How to examine First Failure Support Technology (FFST) files 140
Using trace on AIX 141
Using trace on HP-UX, Solaris and Linux 141
File names for trace files. 142
How to examine FFSTs 142

Appendix A. A review of transaction management. 143

Appendix B. Notices 145
Trademarks 146

Index 149

Sending your comments to IBM . . . 155

Figures

1. Link between a client and server 3
2. WebSphere MQ server connected to clients on different platforms 5
3. Security in a client-server connection 91
4. Message channels between two queue managers 95
5. Client-connection and server-connection on an MQI channel 96
6. Simple channel definition 98
7. Defining the server-connection channel 100
8. Defining the client-connection channel 101
9. MQCONN example 131

Tables

1. The supported external transaction managers for each extended transactional client platform . 11
2. Transmission protocols for MQI channels 13
3. Sets of client components on Client CD 38
4. Order for removing components 41
5. Features installed with each type of installation 46
6. Valid feature names 53
7. Msiexec command line parameters 54
8. Msiexec PROPERTY= value parameters 56
9. Response file parameters 57
10. Supplied transform files 58
11. Properties used by MQParms in the MSI stanza 60
12. Transmission protocols - combination of WebSphere MQ client and server platforms . . 67
13. Maximum outstanding connection requests queued at a TCP/IP port 69
14. Assumed values of the AXLIB parameter 75
15. WebSphere MQ libraries containing the XA switch structures 76
16. The switch load files 77
17. CICS task termination exits 78
18. The location of the com.ibm.mqetclient.jar file 80
19. Programming languages supported in client environments 123
20. Client system libraries on AIX, HP-UX, and Solaris 135
21. Client system libraries on Windows systems 135
22. Sample programs for AIX, HP-UX, and Solaris client systems 136
23. Sample programs for Windows client systems 136

About this book

This book contains information about the WebSphere® MQ client/server environment including extended transactional clients. It describes how to install a WebSphere MQ client, how to configure the communication links and how to set up WebSphere MQ channels so that your WebSphere MQ applications can run on the client machine.

The WebSphere MQ environment variables are described and there are chapters about building and running your applications on a WebSphere MQ client.

Most of the information you need to know about WebSphere MQ clients is in this book. Some of the reference material in the other WebSphere MQ books includes information about WebSphere MQ clients. That reference material is not repeated here.

This book is intended for system administrators, for anyone who installs and configures WebSphere MQ products for the client-server environment, and for application programmers who write programs to make use of the Message Queue Interface (MQI).

This book does not describe the WebSphere MQ Java™ clients. For details of WebSphere MQ classes for Java and WebSphere MQ classes for Java Message Service (JMS), see *WebSphere MQ Using Java*.

What you need to know to understand this book

To understand this book, you should have:

- Experience in installing and configuring the system you use for the server.
- Experience with any client platforms that you will be using.
- Understanding of the purpose of the Message Queue Interface (MQI).

  For further information on MQI, see the following manuals:
  - For MQI call constants, see the *WebSphere MQ Application Programming Reference* manual.
  - For commands and responses, see the *WebSphere MQ Programmable Command Formats and Administration Interface* manual, and for events, see the *Monitoring WebSphere MQ* manual.
- If you are using extended transactional clients, knowledge of the external transaction manager you are using to coordinate WebSphere MQ resources and those of other resource managers.
- Experience of WebSphere MQ programs in general, or familiarity with the content of the other WebSphere MQ publications.

Terms used in this book

The term *UNIX® systems* or *UNIX* denotes the following operating systems:

- AIX
- HP-UX
- Linux®
- Solaris

The terms *Linux systems* or *Linux* denotes:

- Linux (x86 platform)
- Linux (POWER™ platform)
- Linux (zSeries® platform)

The term *Windows®* denotes the following Windows operating systems:

- Windows 2000
- Windows XP
- Windows 2003

The variable *mqmtop* represents the name of the base directory where WebSphere MQ is installed on UNIX and Windows systems.

- On AIX®, the actual name of the directory is **/usr/mqm**
- On other UNIX systems, the actual name of the directory is **/opt/mqm**
- On Windows systems, the default directory is **C:\Program Files\IBM\WebSphere MQ** but you might have chosen to install to a different directory.

The term *WebSphere MQ JMS* means WebSphere MQ classes for Java Message Service.

The term *WebSphere MQ Java* means WebSphere MQ classes for Java and WebSphere MQ classes for Java Message Service combined.

The term *WebSphere MQ extended transactional client* means a WebSphere MQ client that has the extended transactional function.

The term *WebSphere MQ base client* means a WebSphere MQ client that does *not* have the extended transactional function.

How to use this book

Read Chapter 1, “Overview of WebSphere MQ clients,” on page 3 first as a brief introduction. “How do I set up a WebSphere MQ client?” on page 6 gives you a list of tasks that you might need to carry out and guides you through the rest of the book.

Summary of changes

This section describes changes in this edition of *WebSphere MQ Clients*.

This book is a revision of *WebSphere MQ Clients Version 5.3 (GC34–6058–01)*.

It also contains material from *WebSphere MQ Extended Transactional Clients Version 5.3 (SC34–6275–00)*, which is no longer published

Changes for this edition (GC34–6590–00)

- Unsupported platforms are removed
- Environment variables MQIPADRV, MQSSLKEYR, MQSSLFIPS, MQSSLCRYP, and MQSSLRESET added.
- Information about prerequisites on iSeries™ added.
- Prerequisite hardware and software levels changed.
- Information about installing from an electronic software download added.
- A new Linux platform, Linux (POWER platform), is added.
- The two existing Linux platforms are renamed to Linux (x86 platform) and Linux (zSeries platform).
- Additional information is included on advanced installation options on Windows.
- Additional information is included on the use of SSL (Secure Sockets Layer).
- Additional information is included on extended transactional clients.

Changes

Part 1. Introduction to WebSphere MQ clients

Chapter 1. Overview of WebSphere MQ clients . . 3
What is a WebSphere MQ client? 3
How the client connects to the server 4
Client and queue manager on the same machine 4
Clients on different platforms. 4
What is an extended transactional client? 5
Why use WebSphere MQ clients? 6
What applications run on a WebSphere MQ client? 6
How do I set up a WebSphere MQ client? 6

Introduction

Chapter 1. Overview of WebSphere MQ clients

This chapter discusses the following topics:

- "What is a WebSphere MQ client?"
- "Why use WebSphere MQ clients?" on page 6
- "How do I set up a WebSphere MQ client?" on page 6

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.

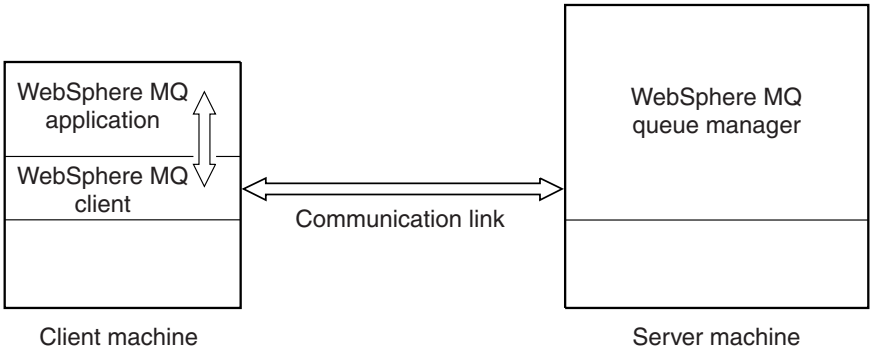

Figure 1. Link between a client and server

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 11.

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 Java or WebSphere MQ classes for Java Message Service (JMS). You can use Java and JMS clients on the i5/OS and z/OS platforms 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 runs in synchronous mode because there must be 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*. When the call succeeds, the MQI channel 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 a full WebSphere MQ 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.

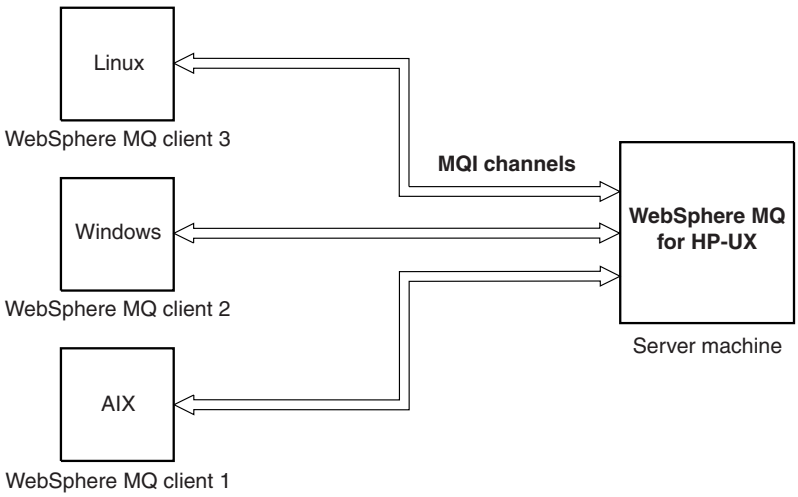

Figure 2. WebSphere MQ server connected to clients on different platforms

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.

What is an extended transactional client?

If you are not familiar with the concepts of transaction management, see Appendix A, "A review of transaction management," on page 143.

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

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 Chapter 13, “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 Chapter 2, “Preparing for installation,” on page 11.
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 Chapter 3, “Installing client components,” on page 21.

3. Ensure that your communication links are configured and connected. This is described in Chapter 4, “Configuring communication links,” on page 67.
4. Check that your installation is working correctly. This is described in Chapter 6, “Verifying the installation,” on page 83.
5. When you have the verified WebSphere MQ client installation, consider whether you need to take any action on security. This is described in Chapter 7, “Setting up WebSphere MQ client security,” on page 91.
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 Chapter 8, “Using channels,” on page 95. There are some additional considerations if you are using SSL. These are described in Chapter 9, “The Secure Sockets Layer (SSL) on WebSphere MQ clients,” on page 107. You might need to use WebSphere MQ *environment variables* to set up the channels. These are described in Chapter 10, “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:
   - Chapter 11, “Using the message queue interface (MQI),” on page 119
   - Chapter 12, “Building applications for WebSphere MQ clients,” on page 123
   - Chapter 13, “Running applications on WebSphere MQ clients,” on page 129
   - Chapter 15, “Solving problems,” on page 139

Why use WebSphere MQ clients?

Part 2. Installing WebSphere MQ clients

Chapter 2. Preparing for installation 11
Platform support for WebSphere MQ clients . . . 11
Platform support for extended transactional clients 11
Applications on WebSphere MQ Version 6.0 clients 13
Communications 13
Performance considerations 13
Hardware and software requirements. 14
AIX client: hardware and software required . . 15
Hardware 15
Software 15
Optional software 15
HP-UX client: hardware and software required 16
Hardware 16
Software 16
Optional software 16
Linux client: hardware and software required . . 17
Hardware 17
Software 17
Optional software 18
Solaris client: hardware and software required. . 19
Hardware 19
Software 19
Optional software 19
Windows client: hardware and software required 20
Hardware 20
Software 20
Optional software 20

Chapter 3. Installing client components 21
Installing a WebSphere MQ client and server system 21
Installing from an electronic software download 22
Installing WebSphere MQ clients on the same machine as the server 22
Uninstalling WebSphere MQ clients 23
Installing on AIX 24
Components for AIX 24
Before installing 24
Creating another file system for the client . . 25
Creating the mqm user ID and group. 25
Custom installation. 26
Installing without SSL support 27
Installing the extended transactional function . . 27
Migrating from an earlier version of WebSphere MQ for AIX 28
Changes to the installation path 28
Changing the national language 29
Translated messages 30
Removing a WebSphere MQ client from AIX . . 30
Migrating to and from the WebSphere MQ SSL support. 30
Uninstalling the extended transactional function 30
Installing on HP-UX 32
Components for HP-UX 32
Before installation 32
Installation 33
Installing the extended transactional function . . 33
Translated messages 34
Removing a WebSphere MQ client from HP-UX 34
Migrating to and from the WebSphere MQ SSL support. 34
Uninstalling the extended transactional function 34
Installing on Linux 36
Components for Linux. 36
Before installation 37
Creating the file systems 37
Creating the mqm user ID and mqm group. . 38
Installation 38
Installing the extended transactional function . . 39
Translated messages 40
Removing the WebSphere MQ client from Linux 40
Migrating to and from the WebSphere MQ SSL support. 41
Uninstalling the extended transactional function 41
Installing on Solaris 42
Components for Solaris 42
Before installation 42
Installation 43
Installing the extended transactional function . . 43
Unattended, or silent, installation 44
Translated messages 44
Removing a WebSphere MQ client from Solaris 45
Migrating to and from the WebSphere MQ SSL support. 45
Uninstalling the extended transactional function 45
Installing on Windows. 46
Preparing to install the WebSphere MQ client . . 46
Installing the WebSphere MQ client 46
Typical client installation 46
Compact client installation 48
Custom client installation. 48
Modifying the client installation 48
Modifying the client installation using Add/Remove Programs 49
Installing the extended transactional function 49
Other methods of installing the WebSphere MQ client 50
Installing from a LAN. 50
Installing the extended transactional client from a LAN server 50
Unattended (silent) installation 51
Advanced installation methods. 53
Using Microsoft System Management Server 61
Installing the extended transactional client using SMS. 61
Uninstalling a WebSphere MQ client 61
Uninstalling WebSphere MQ client from Windows 61
Uninstalling the extended transactional function 63

Chapter 4. Configuring communication links . . 67
Deciding which communication type to use . . . 67
Defining a TCP/IP connection 69
Defining a TCP/IP connection on a WebSphere MQ client 69
Defining a TCP/IP connection on a WebSphere MQ server 69
TCP/IP connection limits 69
Defining an LU 6.2 connection 70
Defining an LU 6.2 connection on a WebSphere MQ client 70
Defining an LU 6.2 Connection on a WebSphere MQ server 70
Defining a NetBIOS connection 70
Defining an SPX connection 70

Chapter 5. Configuring an extended transactional client 71
XA compliant transaction managers 71
The xa_open string 73
The format of an xa_open string 73
How an extended transactional client uses an xa_open string 74
The XA switch structures 76
Dynamic registration 77
Configuring for CICS 77
Configuring for Encina 78
Configuring for Tuxedo 79
Microsoft Transaction Server 79
WebSphere Application Server 80

Chapter 6. Verifying the installation 83
The installation used for the example 83
What the example shows 83
Security 83
Setting up the server on UNIX and Windows systems 83
Setting up a Windows server using WebSphere MQ Explorer 84
Setting up the server on i5/OS 85
Setting up the server on z/OS 85
Setting up the WebSphere MQ client 86
Defining a client-connection channel using MQSERVER 86
Defining a client-connection channel using WebSphere MQ Explorer 86
Putting a message on the queue 87
On the WebSphere MQ client workstation . . . 87
Getting the message from the queue 87
On the WebSphere MQ client workstation . . . 87
Ending verification 87

Chapter 2. Preparing for installation

This chapter discusses the following topics:

- "Platform support for WebSphere MQ clients"
- "Communications" on page 13
- "Hardware and software requirements" on page 14

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 Version 6.0 on the following platforms:

- AIX
- HP-UX
- iSeries
- Linux (x86 platform)
- Linux (POWER platform)
- Linux (zSeries platform)
- Solaris
- 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, apart from Linux (POWER platform). Table 1 lists the supported external transaction managers for each platform.

Table 1. The supported external transaction managers for each extended transactional client platform

Extended transactional client platform	Supported external transaction managers
AIX	BEA Tuxedo, V8.1 TXSeries, V5.1 WebSphere Application Server, V5.1 WebLogic, V8.1
HP-UX	BEA Tuxedo, V8.1 TXSeries, V5.1 WebSphere Application Server, V5.1 WebLogic, V8.1

Table 1. The supported external transaction managers for each extended transactional client platform (continued)

Extended transactional client platform	Supported external transaction managers
Linux (x86 platform)	BEA Tuxedo, V8.1 WebSphere Application Server, V5.1 WebLogic, V8.1
Linux (zSeries platform)	BEA Tuxedo, V8.1 WebSphere Application Server, V5.1
Solaris	BEA Tuxedo, V8.1 TXSeries, V5.1 WebSphere Application Server, V5.1 WebLogic, V8.1
Windows systems	BEA Tuxedo, V8.1 TXSeries, V5.1 WebSphere Application Server, V5.1 Microsoft transaction server (MTS)/COM+ WebLogic, V8.1

A client application that is using an extended transactional client can connect to a queue manager of the following WebSphere MQ Version 6.0 products only:

- WebSphere MQ for AIX
- WebSphere MQ for HP-UX
- WebSphere MQ for iSeries
- WebSphere MQ for Linux (x86 platform)
- WebSphere MQ for Linux (zSeries platform)
- WebSphere MQ for Solaris
- WebSphere MQ for Windows

Even though there is no extended transactional client that runs on i5/OS, a client application that is using an extended transactional client can still connect to a queue manager that runs on i5/OS.

Note, however, that an extended transactional client cannot connect to a queue manager 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.

Client applications that are written in Java and use WebSphere Application Server as the transaction manager can use only WebSphere MQ JMS to access the resources of a queue manager. This is because only WebSphere MQ JMS supports the Java Transaction API (JTA). To support WebSphere MQ JMS applications, a WebSphere MQ base client must include WebSphere MQ Java.

Applications on WebSphere MQ Version 6.0 clients

A Version 6.0 client can connect to all queue managers, non-Version 6.0 as well as Version 6.0. If you are connecting to a non-Version 6.0 queue manager, you cannot use Version 6.0 features and structures in your WebSphere MQ application on the client.

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

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 96.

The transmission protocols possible are shown in the following table:

Table 2. Transmission protocols for MQI channels

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

Notes:

1. Except on Linux (POWER platform)

Table 12 on page 67 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 is synchronous, 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. Channel exits can also be used to compress the data transmitted.

Hardware and software requirements

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

Client platform	Page
AIX	15
HP-UX	16
Linux	17
Solaris	19
Windows	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 section outlines the hardware and software requirements for the WebSphere MQ client for AIX.

Hardware

The WebSphere MQ client for AIX runs on any machine that supports the AIX5L V5.2 or AIX5L V5.3 operating systems capable of running 64-bit programs whether from IBM® or other vendors.There must be enough random access memory (RAM) and disk storage for the programming prerequisites specified later in this section, the WebSphere MQ client, the access methods, and the application programs.

Software

The following are the software prerequisites for the WebSphere MQ client for AIX. Minimum supported levels are shown. Later levels, if any, are supported unless otherwise stated.

Operating system:

- AIX V5.2 (64–bit).
- AIX V5.3 (64 bit).

Use the **oslevel** command to determine the level of the operating system you are running.

Connectivity:

- TCP/IP (as part of the base operating system)
- IBM Communications Server for AIX, Version 6.1 (for SNA LU 6.2 connectivity)

Optional software

Secure Sockets Layer (SSL): If you want to use the WebSphere MQ SSL support, you need the IBM Global Security Kit (GSKit), Version 7, which is supplied with WebSphere MQ.

Compilers for WebSphere MQ client applications on an AIX system: The following compilers are supported:

- IBM Software Developer's Kit (SDK), Java 2 Technology Edition for AIX, Version 1.4.2
- IBM C for AIX, V5.0
- IBM C for AIX, V6.0
- IBM C for AIX, V7.0
- IBM Visual Age C++ Professional for AIX, V5.0
- IBM Visual Age C++ Professional for AIX, V6.0
- IBM Visual Age C++ Professional for AIX, V7.0
- IBM COBOL Set for AIX, V1.1 (for 32-bit applications only
- Micro Focus Server Express V4.0

HP-UX client: hardware and software required

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

Hardware

The WebSphere MQ client for AIX runs on any Hewlett Packard PA-RISC 2.0 machine that supports the specified operating system.

Software

The following are the software prerequisites for the WebSphere MQ client for HP-UX. Minimum supported levels are shown. Later levels, if any, are supported unless otherwise stated.

Operating system:

- HP-UX 11i v1 (B.11.11) (plus Dec. 2003 QPK)
- HP-UX 11i v2 (11.23) for IPF

Connectivity:

- TCP/IP (as part of the base operating system)
- HP SNAplus2, Version 6 (for SNA LU 6.2 connectivity)

Optional software

Secure Sockets Layer (SSL): If you want to use the WebSphere MQ SSL support, you need the IBM Global Security Kit (GSKit), Version 7, which is supplied with WebSphere MQ.

Compilers for WebSphere MQ client applications on an HP-UX system: The following compilers are supported:

- HP-UX Software Development Kit for the Java 2 Platform, Version 1.4.2
- HP SDK for J2SE HP-UX 11i V2 platform, adapted by IBM for IBM Software, V1.4.2
- Micro Focus Server Express,V4.0
- HP C/ANSI C Developer's Bundle for HP-UX 11.0 and11i Version 1
- HP C/ANSI C Developer's Bundle V5.60 (for HP-UX 11i v2)
- HP aC++ version A.03.52 for HP-UX 11.0 and11i Version 1
- HP aC++ V6.0 (for HP-UX 11i v2)

Linux client: hardware and software required

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

Hardware

The WebSphere MQ client for Linux (x86 platform) runs on any machine that supports the x86 machine architecture. The WebSphere MQ client for Linux (POWER platform) runs on any machine that supports the POWER machine architecture. The WebSphere MQ client for Linux (zSeries platform) runs on any machine that supports the zSeries machine architecture.

For connectivity, you can use any communications hardware that supports SNA LU 6.2 (Linux (x86 platform) only) or TCP/IP.

Software

The following are the software prerequisites for the WebSphere MQ client for Linux. Minimum supported levels are shown. Later levels, if any, are supported unless otherwise stated.

Operating system: The WebSphere MQ client for Linux can be installed on any distribution of Linux (x86 platform), Linux (zSeries platform) or Linux (POWER platform) that supports:

- Linux kernel, Version 2.4.21
- glibc, Version 2.2
- pthreads, Version 0.7
- libstdc++, Version 3.0, for C++ programming
- Red Hat Package Manager (RPM), for installation

The WebSphere MQ client for Linux has been tested with the following distributions of Linux (x86 platform):

- Red Hat Enterprise Linux 3.0 (plus Update 2)
- Red Hat Enterprise Linux 4.0
- SUSE LINUX Enterprise Server (SLES) 8 (plus Service Pack 3)
- SUSE LINUX Enterprise Server (SLES) 9

The WebSphere MQ client for Linux has been tested with the following distributions of Linux (zSeries platform):

- Red Hat Enterprise Linux 3.0 (plus Update 2)
- Red Hat Enterprise Linux 4.0
- SUSE LINUX Enterprise Server (SLES) 8 (plus Service Pack 3)
- SUSE LINUX Enterprise Server (SLES) 9

The WebSphere MQ client for Linux has been tested with the following distributions of Linux (POWER platform):

- SUSE LINUX Enterprise Server (SLES) 9
- Red Hat Enterprise Linux , V3.0
- Red Hat Enterprise Linux , V4.0

Connectivity:

- IBM Communications Server for Linux, Version 6.2 (except Linux (zSeries platform))

- IBM Communications Server for Linux on zSeries, Version 6.2 (Linux (zSeries platform) only)
- TCP/IP, as part of the base operating system

Optional software

Secure Sockets Layer (SSL): If you want to use the WebSphere MQ SSL support, you need the IBM Global Security Kit (GSKit), Version 7, which is supplied with WebSphere MQ.

Compilers for WebSphere MQ client applications on a Linux system: The following compilers are supported:

- GNU C Compiler (gcc) and g++, Version 2.9.5 (not applicable to Linux (POWER platform))
- GNU C Compiler (gcc) and g++, Version 3.2
- GNU C Compiler (gcc) and g++, Version 3.3
- GNU C Compiler (gcc) and g++, Version 3.4
- IBM Software Developer's Kit (SDK), Java 2 Technology Edition for Linux, Version 1.4.2 (Linux (x86 platform) only)
- IBM Software Developer's Kit (SDK), Java 2 Technology Edition for Linux on POWER, Version 1.4.2
- IBM Software Developer's Kit (SDK), Java 2 Technology Edition for Linux on zSeries, Version 1.4.2
- Micro Focus Server Express,V4.0

Solaris client: hardware and software required

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

Hardware

The WebSphere MQ client can run on all Sun SPARC and Sun UltraSPARC desktop and server systems, supported by the appropriate release of the Solaris operating environment.

For connectivity, you can use any communications hardware that supports SNA LU 6.2 or TCP/IP.

Software

The following are the software prerequisites for the WebSphere MQ client for Solaris. Minimum supported levels are shown. Later levels, if any, are supported unless otherwise stated.

Operating system:

- Solaris 8 (plus SunSolve recommended Patch Cluster level)
- Solaris 9 (plus SunSolve recommended Patch Cluster level)

Connectivity:

- TCP/IP (as part of the base operating system)
- Data Connection SNAP-IX, Version 7.0

Optional software

Secure Sockets Layer (SSL): If you want to use the WebSphere MQ SSL support, you need the IBM Global Security Kit (GSKit), Version 7, which is supplied with WebSphere MQ.

Compilers for WebSphere MQ client applications on a Solaris system: The following compilers are supported:

- Sun ONE Studio 7, Enterprise Edition for Solaris
- Sun ONE Studio 8, Compiler Collection
- Micro Focus Server Express,V4.0
- Sun's Java 2 SDK, Enterprise Edition, Version 1.4

Windows client: hardware and software required

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

Hardware

The WebSphere MQ client can run on any IBM PC machine, or equivalent, certified as Windows compatible, and capable of running Windows 2000, Windows XP or Windows 2003. There must be enough random access memory (RAM) and disk storage for the programming prerequisites specified later in this section, the WebSphere MQ client, the access methods, and the application programs.

Software

The following are the software prerequisites for the WebSphere MQ client for Windows. Minimum supported levels are shown. Later levels, if any, are supported unless otherwise stated.

Operating system:

- Microsoft® Windows 2000 with Service Pack 4. This can be any of the following products:
  - Microsoft Windows 2000 Professional
  - Microsoft Windows 2000 Server
  - Microsoft Windows 2000 Advanced Server
- Microsoft Windows XP Professional
- Microsoft Windows 2003 Server

Connectivity:

- IBM Communications Server for Windows, Version 6.1.2
- IBM Personal Communications, Version 5.7(Part of IBM Host access client package (HACP) V4.0
- Microsoft Host Integration Server 2004
- Microsoft Host Integration Server 2000
- Attachmate myEXTRA! Presentation Services, Version 7.11
- Attachmate Extra! Personal Client, Version 6.7
- Attachmate Extra! Enterprise 2000
- TCP/IP, NetBIOS, and SPX, as part of the base operating system

Optional software

Compilers for WebSphere MQ client applications on a Windows system: The following compilers are supported:

- Microsoft Visual C++ .NET 2003
- Microsoft Visual C# .NET 2003
- Microsoft Visual Basic .NET 2003
- Microsoft Visual Basic V6
- IBM VisualAge® COBOL Enterprise, V3.0.1
- Micro Focus Net Express, V4.0
- IBM Software Developer's Kit (SDK), Java 2 Technology Edition for Windows, Version 1.4.2

Other:

- Microsoft Windows Terminal Server feature

Chapter 3. Installing client components

This chapter discusses how to install the client components for WebSphere MQ Version 6.0 on UNIX systems and on Windows.

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 on AIX" on page 24
- "Installing on HP-UX" on page 32
- "Installing on Linux" on page 36

- "Installing on Solaris" on page 42
- "Installing on Windows" on page 46

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 14).

MQ60Client_solaris.tar	120MB
MQ60ClientSSL_solaris.tar	20MB
MQ60Client_hpux.tar	70MB
MQ60ClientSSL_hpux.tar	190MB
MQ60Client_aix.tar	40MB
MQ60ClientSSL_aix.tar	110MB
MQ60Client_LinuxIntel.tar	250MB
MQ60ClientSSL_LinuxIntel.tar	310MB
MQ60Client_LinuxzSeries.tar	85MB
MQ60ClientSSL_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 can 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 Chapter 8, "Using channels," on page 95 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 on AIX

To install the WebSphere MQ client on an AIX system, use WebSphere MQ Client CD 1, which is supplied with WebSphere MQ.

The WebSphere MQ client is installed into the **/usr/mqm** directory. This ***cannot*** be changed. However, if you do not have enough space in the **/usr/mqm** file system, follow the procedure given in "Creating another file system for the client" on page 25.

If you have an earlier version than WebSphere MQ Version 6.0 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" on page 28.

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 22.

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 Java messaging 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.

Before installing

Before you can install the WebSphere MQ client on your AIX system you are advised to create and mount a `/var/mqm` file system, or `/var/mqm`, `/var/mqm/log`, and `/var/mqm/errors` file systems.

If you create separate filesets, the following directories ***must*** be on a local file system:
- `/var/mqm`
- `/var/mqm/log`

You can choose to NFS mount the `/var/mqm/errors` directory to conserve space on your local system.

A user ID with the name `mqm`, whose primary group is `mqm`, is automatically created during installation. You can create the user and group IDs yourself (see "Creating the mqm user ID and group" on page 25), but make sure you do this before installing the client. User ID and group must both be `mqm`.

For stand-alone machines, you can create the new user and group IDs locally. For machines administered in a network information services (NIS) domain, you can create the user and group IDs on the NIS master server machine.

After installation, the mqm user ID owns the directories and files that contain the resources associated with the product. This group and user must be defined for any machine on which the WebSphere MQ software is to be installed, whether the machine is a client or a server machine.

Creating another file system for the client

If you do not have enough space in the **/usr/mqm** file system, you can do either of the following things:

1. Create a new file system and mount it as **/usr/mqm**.

or

2. Create a new directory anywhere on your machine that is large enough to contain the client files, and create a symbolic link from **/usr/mqm** to this new directory. For example:

```
mkdir /bigdisk/mqm
ln -s /bigdisk/mqm /usr/mqm
```

Whichever of these options you choose, you ***must*** do it before installing the client.

The file system into which the client is installed can be a remote network device, for example NFS, provided that the mount options are defined on that device to allow setuid programs (including root access) to be run.

Creating the mqm user ID and group

If you want to create the required IDs yourself, for instance, if you want to set up all security groups before installing client components, follow this procedure before you install the client. You must create both user ID and group as mqm.

Create the new IDs using the System Management Interface Tool (SMIT), for which you require root authority. The procedure for this, if you use the SMIT windows, is:

1. Create the mqm group. You can display the required window using this sequence:

```
Smit
   Security & Users
      Groups
         Add a Group
```

   Set the name field to mqm

2. Create the new user, mqm. You can display the window for doing this using this sequence:

```
Smit
   Security & Users
      Users
         Add a User
```

   Set the name field to mqm

   Set the primary group for this user to be mqm. You can take the default values for the attributes of the new group or change them if you wish.

3. Add a password to the new user ID. You can display the window for doing this using this sequence:

```
Smit
   Security & Users
      Change a user's Password
```

If you are entering a new user `mqm`, the old password is `mqm`

4. Add the newly created group `mqm` to an existing user ID. You can display the window for doing this using this sequence:

```
Smit
   Security & Users
      Users
         Change / Show Characteristics of a User
```

When the window is displayed, enter the name of the user who is to have the `mqm` group added. In the user name field, add `mqm` to the **Group SET** field, which is a comma-separated list of the groups to which the user belongs.

Note: You need not have your primary group set to `mqm`. As long as `mqm` is in your set of groups, you can use the commands. If you are running applications that use the queue manager only, you do not need `mqm` group authority.

Custom installation

This section describes custom installation using the System Management Interface Tool (SMIT). By default, the procedure installs the client components for the WebSphere MQ SSL support. If you do not want to install the client components for the WebSphere MQ SSL support, see "Installing without SSL support" on page 27.

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" on page 28.

You can use SMIT for a custom installation as follows:

1. Log on as root.
2. Go into SMIT and from the shell, type:

   ```
   smit
   ```

3. Select the device appropriate for your installation, using this sequence of windows:

   ```
   Software Installation and Maintenance
      Install and Update Software
         Install and Update from Latest Available Software
   ```

   Alternatively, you can use the fastpath command:

   ```
   smitty install_latest
   ```

4. Press the **List** button to display the **Single Select List** window.
5. Select **/dev/cd0 (CD Drive)**
6. Press **Do** to display the parameters for **Install Latest Level**.
7. Press **F4** to get a list of filesets to install.
8. Follow the SMIT instructions to select the components you want to install.
9. Press **Enter**.
10. If you have a previous version of the product on your machine, change the **Automatically install requisite software** to **No** and **Overwrite same or newer versions** to **Yes**.
11. On AIX V4.3.3:

- Press **OK** to install.

On AIX V5.1

a. Change "Preview new LICENSE agreements?" to Yes, and click **OK** to view the license agreements.
b. Change "ACCEPT new license agreements?" to Yes, and click **OK** to accept the license agreements and install WebSphere MQ.

Now go to Chapter 6, "Verifying the installation," on page 83.

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" on page 28.

Follow the instruction in "Custom installation" on page 26 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

If you want to migrate from an earlier version of a WebSphere MQ for AIX client to an IBM WebSphere MQ for AIX, Version 6.0 client, you must first end all WebSphere MQ activity on the target machine, and remove any shared resources that are used by WebSphere MQ. Do this either by shutting down the system and restarting it, or by issuing the **icprm** command to remove the shared resources.

The migration procedure described in this section applies only to migration from an earlier version of a WebSphere MQ for AIX client to IBM WebSphere MQ for AIX, Version 6.0 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 6.0 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.

Once all previously installed filesets have been updated to the latest level, you can install any additional filesets. See “Custom installation” on page 26 for more information.

Changes to the installation path

Changes in AIX LPP Version 4 packaging mean that the IBM WebSphere MQ for AIX, Version 6.0 client installs 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, Version 6.0 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, Version 6.0 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, Version 6.0 client as described in "Custom installation" on page 26.

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" on page 28.

Note: If you have already symbolically linked a file system to **/usr/lpp/mqm**, installation of the IBM WebSphere MQ for AIX, Version 6.0 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, Version 6.0 client.

The installation process for the IBM WebSphere MQ for AIX, Version 6.0 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

The easy installation and the custom installation default 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. Use SMIT as follows to install the message catalog for another national language:

1. Type `smit`
2. Follow this sequence of windows:

   ```
   Software Installation and Maintenance
     Install and Update Software
       Install and Update from ALL Available Software
   ```

3. Press the **List** button to display the **Single Select List** window.
4. Select:
   /dev/cd0 (CD Drive)
5. Press the **List** button on the **Software to Install** field.
6. Select the message catalog that you want to install.
7. Press **OK** to install the chosen message catalog or catalogs.

Translated messages

Messages in U.S. English are always available. If you require one of the other languages that is supported by WebSphere MQ for AIX, you ***must*** ensure that your NLSPATH environment variable includes the appropriate directory.

For example, to select messages in German use the following:

```
export LANG=de_DE
export NLSPATH=/usr/lib/nls/msg/%L/%N
```

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 instructions in “Custom installation” on page 26, choosing only those file sets. To downgrade a WebSphere MQ client with the SSL support to one without the SSL support, simply 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.

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 22.

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 Java messaging 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.

Before installation

Before you can install a WebSphere MQ client on your HP-UX system you:

- Must create a group with the name `mqm`.
- Must create a user ID with the name `mqm`.
- Are recommended to create and mount a `/var/mqm` file system, or `/var/mqm`, `/var/mqm/log`, and `/var/mqm/errors` file systems.

  If you create separate partitions, the following directories ***must*** be on a local file system:
  - `/var/mqm`
  - `/var/mqm/log`

  You can choose to NFS mount the `/var/mqm/errors` and `/var/mqm/trace` directories to conserve space on your local system.

After installation, this user ID (mqm) owns the directories and files that contain the resources associated with the product. This group and user must be defined for any machine on which the WebSphere MQ software is to be installed, whether the machine is a client or a server machine.

For stand-alone machines, you can create the new user and group IDs locally. For machines administered in a network information services (NIS) domain, you can create the user and group IDs on the NIS master server machine.

Installation

Use the HP-UX **swinstall** program, or use SAM, after mounting the CD. For further details, see the appropriate HP-UX documentation.

1. Log in as root.
2. Insert WebSphere MQ Client CD 2 into the CD drive.
3. Mount the CD drive
   a. Change directory to /usr/$bin
   b. Type `pfs_mountd &`
   c. Type `pfsd4 &`
   d. Type `pfs_mount –o xlat=unix /<path to CD device>/<localdir>`
4. Accept the licence:
   a. Change directory to the location of the mounted CD (for example, `/cdrom`).
   b. Run the mqlicense.sh script by typing one of the following commands:
      - For the WebSphere MQ client without the WebSphere MQ SSL support:
        `./hpux11/MQClient/mqlicense.sh`
      - For the WebSphere MQ client with the WebSphere MQ SSL support:
        `./hpux11/MQClientwithSSL/mqlicense.sh`

      The license is displayed. If you accept the license, you can continue the installation. If you decline, the message: **The license agreement has not been accepted ...** is displayed, and the installation will fail.

      If you are performing a silent or remote install, you can run the installation with the `-accept` option, so that the license is accepted without being displayed.
5. Type one of the following commands to start the installation procedure:
   - For the WebSphere MQ client without the WebSphere MQ SSL support:
     `swinstall -s /cdrom/<localdir>/hpux11/MQClient/mqs530.v11`
   - For the WebSphere MQ client with the WebSphere MQ SSL support:
     `swinstall -s /cdrom/<localdir>/hpux11/MQClientwithSSL/mqs530.v11`

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 /cdrom/hpux11/mqs530.v11 MQSERIES.MQM-TXCLIENT
   ```

   The installation now runs to completion.

Translated messages

Messages in U.S. English are always available. If you require one of the other languages that is supported by WebSphere MQ for HP-UX, you ***must*** ensure that your NLSPATH environment variable includes the appropriate directory.

For example, to select messages in German use the following:

```
export LANG=de_De.iso88591
export NLSPATH=/usr/lib/nls/msg/%L/%N
```

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 instructions in “Installation” on page 33, selecting only those file sets. To downgrade a WebSphere MQ client with the SSL support to one without the SSL support, simply 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.

The WebSphere MQ client is installed in the `/opt/mqm` directory. You *cannot* change this directory. If you do not have enough space in the `/opt/mqm` file system, follow the procedure in "Creating a file system for the components" on page 37.

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 22.

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 Java messaging components of the WebSphere MQ base client.

Components for Linux

You can install the following components on a Linux client system:

Client The WebSphere MQ client code for the Linux platform.

Runtime
This component provides support for applications. You must install this component before you install the Client component.

SDK This component is needed for application development.

Sample programs
Sample application programs that are needed if you want to verify that the WebSphere MQ client has been installed correctly.

Message catalogs
A message catalog in US English is installed automatically. Message catalogs are also available for the following national languages:

- Brazilian Portuguese
- French
- German
- Italian
- Japanese
- Korean
- Spanish
- Simplified Chinese
- Traditional Chinese

IBM Global Security Kit (GSKit) V7
This component is needed for the WebSphere MQ SSL support. GSKit contains the key management tool, iKeyman.

WebSphere MQ support for iKeyman
This component is needed for the WebSphere MQ SSL support. You must install this component after you have installed the IBM Global Security Kit (GSKit).

Before installation

Before you install the WebSphere MQ client, you must do the following:

1. Create the file systems that are used to store the components and working data.
2. Create the mqm user ID and mqm group.

Creating the file systems

The components are installed in /opt/mqm and working data is stored in /var/mqm. You cannot change these directories.

Creating a file system for the components: The disk space required for the components depends on how many components you install. Typical disk space requirements are as follows:

Linux (x86 platform)

- Without the SSL support: 27 MB
- With the SSL support: 100 MB

Linux (zSeries platform)

- Without the SSL support: 28 MB
- With the SSL support: 93 MB

Linux (POWER platform)

- Without the SSL support: ?? MB
- With the SSL support: ?? MB

If you do not have enough space to store the components in the /opt/mqm file system, you can do either of the following:

- Create a new file system and mount it as /opt/mqm.
- Create a new directory anywhere on your machine, and create a symbolic link from /opt/mqm to this new directory. For example:

```
mkdir /bigdisk/mqm
ln -s /bigdisk/mqm /opt/mqm
```

Whichever of these options you choose, you must do it *before* you install the WebSphere MQ client.

The file system for the components can be a remote network device, using NFS, for example. However, the mount options defined on that device must allow **setuid** programs, including root access, to run.

Creating a file system for the working data: Before you install the WebSphere MQ client, create and mount a file system called /var/mqm. In this way, other system activity is not affected if a large amount of working data builds up.

To determine the size of the /var/mqm file system for a client, consider:

- The size of the error log files in the /var/mqm/errors directory
- The amount of trace data that is written to the /var/mqm/trace directory

Typically, allow 15 MB of disk space in the /var/mqm file system.

Creating a separate file system for the error log files: You can also create a separate file system, /var/mqm/errors, for the error log files. This file system can be NFS mounted but, if you choose to do this, you might lose the error logs if the network fails. Typically, allow 4 MB of disk space for the error log files.

Creating the mqm user ID and mqm group

The mqm user ID, with a primary group of mqm, is created automatically during the installation of WebSphere MQ, unless you are using Caldera OpenLinux. After installation, the mqm user ID owns the directories and files that contain the resources associated with the product.

You can create the mqm user ID and mqm group yourself. You might do this, for example, if you want to configure your security environment before installing WebSphere MQ. If you are installing on Caldera OpenLinux, you *must* create the mqm user ID and mqm group yourself *before* installing WebSphere MQ.

Installation

This installation procedure uses the Red Hat Package Manager (RPM) installer, which allows you to choose the components you want to install. The components are listed in "Components for Linux" on page 36.

1. Log in as root.
2. Mount WebSphere MQ Client CD 3 on the target machine.
3. Select the set of client components that you want to use for the installation, and change to the corresponding directory on the CD.

   Table 3 shows each set of client components and its corresponding directory on the CD.

Table 3. Sets of client components on Client CD

Set of client components	Directory on WebSphere MQ Client CD
Linux (x86 platform)	`/linux_intel/MQClient`
Linux (zSeries platform)	`/linux_zseries/MQClient`
Linux (POWER platform)	

Notes:

a. If you do not have a locally attached CD drive, you can copy the contents of the directory you require from a machine that does have a CD drive to the target machine using, for example, the ftp utility. If you do this, ensure that you copy the entire directory structure for the set of client components you have selected, and that you maintain the same directory structure on the target machine.

   You can now install the client components from the local copy of the directory.

b. If the machine hosting the CD is an NFS server, you can mount the contents of the CD on the target machine using NFS.

4. Run the `mqlicense.sh` script.

   The license is displayed. If you accept the license, the installation continues. If you decline, the message:

   ```
   Product cannot be installed until the license agreement has been accepted...
   ```

   is displayed, and the installation fails.

   If you are performing a silent or remote install, you can run the `mqlicense.sh` script with the `-accept` option, so that the license is accepted without being displayed.

5. Use the **rpm -i** command to install a minimum client configuration. For example:

- If you are installing on an Intel® machine, enter the following commands, in the order shown, to install the Runtime and Client components:

```
rpm -i MQSeriesRuntime-5.3.0-1.i386.rpm
rpm -i MQSeriesClient-5.3.0-1.i386.rpm
```

- If you are installing on a zSeries machine, enter the following commands, in the order shown, to install the Runtime and Client components:

```
rpm -i MQSeriesRuntime-5.3.0-1.s390.rpm
rpm -i MQSeriesClient-5.3.0-1.s390.rpm
```

The WebSphere MQ license notice is shown only when the first component is installed. It is not shown again for subsequent components.

6. Use the **rpm -i** command to install any other components that you require.

   For example, to install the IBM Global Security Kit (GSKit) and the WebSphere MQ support for iKeyman on a zSeries machine, enter the following commands in the order shown:

```
rpm -i gsk6bas-6.0-n.nn.s390.rpm
rpm -i MQSeriesKeyman-5.3.0-1.s390.rpm
```

   Look for the `gsk6bas` package on the CD to find the value of *n.nn*.

Note: Distributions that do not use the Red Hat Package Manager (RPM) installer by default might generate an error when you install the WebSphere MQ 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

   Notes:

   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 -i** command to install the package containing the extended transactional function:
   - If you are installing on an Intel machine, enter the following command:
     ```
     rpm -i MQSeriesTXClient-5.3.0-2.i386.rpm
     ```
   - If you are installing on a zSeries machine, enter the following command:
     ```
     rpm -i MQSeriesTXClient-5.3.0-2.s390.rpm
     ```

Translated messages

Messages in U.S. English are always available. If you require messages in a different language, ensure that:

1. You install the appropriate message catalog. See "Components for Linux" on page 36.
2. Your NLSPATH environment variable includes the appropriate directory. For example, to select messages in German use the following:
   ```
   export LANG=de
   export NLSPATH=/usr/share/locale/%L/LC_MESSAGES/%N
   ```

To find out which language is currently installed, use the **locale** command.

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 on page 41. 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]

Notes:

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 in "Installation" on page 38, selecting only those two components. To downgrade a WebSphere MQ client with the SSL support to one without the SSL support, simply 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 40.

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.

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 22.

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.

Before installation

Before you can install a WebSphere MQ client on your Solaris system you:

- Must create a group with the name `mqm`.
- Must create a user ID with the name `mqm`.
- Are recommended to create and mount a `/var/mqm` file system, or `/var/mqm`, `/var/mqm/log`, and `/var/mqm/errors` file systems.

  If you create separate partitions, the following directories ***must*** be on a local file system:
  - `/var/mqm`
  - `/var/mqm/log`

  You can choose to NFS mount the `/var/mqm/errors` and `/var/mqm/trace` directories to conserve space on your local system.
- Must set the umask of the root userid to 022. The installation can fail if other settings are used.

After installation, this user ID (`mqm`) owns the directories and files that contain the resources associated with the product. This group and user must be defined for any machine on which the WebSphere MQ software is to be installed, whether the machine is a client or a server machine.

For stand-alone machines, you can create the new user and group IDs locally. For machines administered in a network information services (NIS) domain, you can create the user and group IDs on the NIS master server machine.

Installation

Note: The WebSphere MQ product is installed into the **/opt/mqm** directory. This ***cannot*** be changed.

Carry out the following procedure:

1. Check whether the Volume Manager is running on your system by typing the following command:

   ```
   /usr/bin/ps -ef | /bin/grep vold
   ```

   If the Volume Manager is running, the CD is mounted on /cdrom/MQ60Client1 automatically. If it is not running, mount the CD by typing the following commands:

   ```
   mkdir -p /cdrom/MQ60Client1
   mount -F hsfs -r /dev/dsk/<cntndnsn> /cdrom/MQ60Client1
   ```

   substituting <cntndnsn> with the name of your CD drive.
2. Run the mqlicense.sh script to accept the license by typing one of the following commands:
   - For the WebSphere MQ client without the WebSphere MQ SSL support:

     ```
     /cdrom/MQ60Client1/solaris/MQClient/mqlicense.sh
     ```
   - For the WebSphere MQ client with the WebSphere MQ SSL support:

     ```
     /cdrom/MQ60Client1/solaris/MQClientwithSSL/mqlicense.sh
     ```

   The license is displayed. If you accept the license, the installation continues. If you do not accept the license, you cannot continue the installation process.
3. Use the Solaris **pkgadd** program to install the WebSphere MQ client software by carrying out the following procedure:
   a. Type one of the following commands:
      - For the WebSphere MQ client without the WebSphere MQ SSL support:

        ```
        pkgadd -d /cdrom/MQ60Client1/solaris/MQClient/mqs600.img
        ```
      - For the WebSphere MQ client with the WebSphere MQ SSL support:

        ```
        pkgadd -d /cdrom/MQ60Client1/solaris/MQClientwithSSL/mqs600.img
        ```

   b. You are prompted for a list of components to be installed. Select the ones you require, If you want to install all the components, select **all**.
   c. Press the Enter key.

For further information on using **pkgadd** to install software packages, see the Solaris documentation.

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 44 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 the following command:

   ```
   pkgadd -d /cdrom/solaris/mqs530.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.

Translated messages

Messages in U.S. English are always available. If you require one of the other languages supported by WebSphere MQ for Solaris, you *must* ensure that your `NLSPATH` environment variable includes the appropriate directory.

For example:

```
export LANG=de
export NLSPATH=/usr/lib/locale/%L/LC_MESSAGES/%N
```

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 "Installation" on page 43). 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."

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"
- "Installing the WebSphere MQ client"
- "Installing from a LAN" on page 50
- "Unattended (silent) installation" on page 51
- "Advanced installation methods" on page 53
- "Using Microsoft System Management Server" on page 61
- "Uninstalling a WebSphere MQ client" on page 61

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, you can 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 • 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 • 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.

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, V6.0 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 48.
   b. If you select Remove, see "Uninstalling WebSphere MQ client using the installation process" on page 62.

   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 46 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 48.
   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 48 from step 3 or 4 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 Chapter 6, "Verifying the installation," on page 83).

Compact client installation

Follow the steps for a typical client installation, as described in "Typical client installation" on page 46. The only difference is that, at step 5 on page 47, 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 to 5 of the "Typical client installation" on page 46.
2. At step 5, 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 47 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 Program Maintenance panel is displayed.

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

1. From the Windows task bar, 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:

1. Insert the WebSphere MQ Extended Transactional Clients 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. In the list of national languages, click the language that you want to use, and then click **OK**. Eventually, the Welcome window opens.
3. Click **Next** to continue. The License Agreement window opens.
4. Read the license agreement.

   To change the language in which the license agreement is displayed, click **Change Language**, select your preferred language from the list provided, and then click **OK**.

Select the option to accept the terms of the license agreement, and click **Next**. The Ready to Install window opens.

5. Click **Install**. A window opens asking whether you have purchased sufficient license units to install the extended transactional client. If you have purchased sufficient license units, click **Yes**. The Installing window opens.
6. Wait for the progress bar to complete.

   When the installation completes successfully, the WebSphere MQ Extended Transactional Client Setup window displays the following message:

   ```
   Installation Wizard Completed Successfully
   ```
7. 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 49 from step 2 on page 49 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.

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 Server CD, or a copy of the files on it, and you must be able to execute a command on that machine.

Notes:

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 "Unattended (silent) installation" on page 51.
2. There are several other methods to invoke MSI without setup.exe. For details, see "Advanced installation methods" on page 53.

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 *WebSphere MQ for Windows, V6.0 Quick Beginnings*.

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
```

where:

/q Specifies a silent installation.

`TRANSFORMS=:1033` specifies that the installation is in US English. For further information about installing in different national languages, see *WebSphere MQ LotusScript Extension*.

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 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, it must be enclosed in double quotation marks.
- For a property that can take more than one value, use the format:

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

See *WebSphere MQ for Windows, V6.0 Quick Beginnings* for details of the features that can be values for the ADDLOCAL and REMOVE properties.

Table 6. Valid feature names

Feature Name	Description
Client	The WebSphere MQ for Windows client.
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"
```

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.

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 AGREETOLICENSE="yes"
```

where /q requests an unattended installation.

TRANSFORMS=:1033 specifies that the installation language is US English. For more information about installing in other national languages, see *WebSphere MQ for Windows, V6.0 Quick Beginnings*.

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

You cannot use a response file.

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: 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 54 shows the parameters you can use. For a silent installation, this must include the **/q** or **/qn** parameter.

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" ADDLOCAL="Client" REMOVE=""
```

Table 8 on page 56 and Table 9 on page 57 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. They can be enclosed in quotation marks. If a value string includes a blank, it must be enclosed in quotation marks.
- For a property that can take more than one value, use the format: `ADDLOCAL="Client,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

Table 7. Msiexec command line parameters (continued)

Parameter	Options	Description
/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** Non-fatal warnings **e** All error messages **a** Start up of actions **r** Action-specific records **u** User requests **c** Initial user interface parameters **m** Out-of-memory or fatal 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"
/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 58.
SAVEINI	*path\file_name*	Generate a response file during installation. The file will contain those parameters selected for this installation that a user could 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).

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*"
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 53.

Using transforms: 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, WebSphere MQ.msi.

On the Msiexec command line, you can specify the required language by using the TRANSFORMS property in a property=value pair, for example: `TRANSFORMS=:1033`

The **:** (colon) character means use the embedded transform. Otherwise, you must specify the full path and file name of the transform file, for example:
`TRANSFORMS=D:\Msi\1033.mst`

Table 10 shows the supplied transform files, the resulting language, and the numerical value to use in the Msiexec command line.

You can also specify the required language by using the MQPLANGUAGE property with the MQParms command. See Table 8 on page 56.

Table 10. Supplied transform files

Language	Transform File name	Value
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

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 51. 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 53 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.

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 54, 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, Response.ini.

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.

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 MQParms.ini. For further details, see "Parameter file" on page 60.

   parameters
   : are one or more command line parameters, as listed in Table 7 on page 54.

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 54, Table 8 on page 56, Table 9 on page 57, and Table 11.

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,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 57, Table 7 on page 54, and Table 8 on page 56 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 58.
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.

Table 11. Properties used by MQParms in the MSI stanza (continued)

Property	Values	Description
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.

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.

- Use the Add/Remove Programs facility in the Windows Control Panel.
- Perform a removal from the command line.

You can use these methods to uninstall the WebSphere MQ client, as long as 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 53.

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 Program Maintenance panel is displayed.

   If this panel is not displayed, WebSphere MQ for Windows, V6.0 is not installed on this machine.
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:

1. From the Windows task bar, select **Start→ Settings→ Control Panel**.
2. Select **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 to the end.

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

  This command invokes a silent uninstall of all features.
- `Msiexec /x "path\MSI\IBM WebSphere MQ.msi"`

  This command displays only a progress dialog whilst uninstalling all features.
- `Msiexec /x "path\MSI\IBM WebSphere MQ.msi" /q`

  This command invokes a silent uninstall of all features.

Alternatively, you can 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 51.

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,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 61.

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

- Start the installation process from the WebSphere MQ Extended Transactional Clients CD-ROM. This gives you the option to uninstall the extended transactional client.
- Use Add/Remove Programs in the Control Panel.
- 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.

1. Insert the WebSphere MQ Extended Transactional Clients 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 root folder on the CD-ROM to start the installation process.

   The Welcome window opens. If this window does not open, the extended transactional client is not installed on your system.
2. Click **Next** to continue. The Remove the WebSphere MQ Extended Transactional Client window opens.
3. Click **Remove**. The Removing the WebSphere MQ Extended Transactional Client window opens.
4. Wait for the progress bar to complete.

   If there are any messages that state that locked files are found, make sure that no WebSphere MQ client applications are running. After you have stopped any client applications, uninstallation should then continue.

   When the uninstallation completes successfully, the WebSphere MQ Extended Transactional Client Setup window displays the following message:

   ```
   Uninstallation Completed Successfully
   ```
5. Click **Finish** to close the WebSphere MQ Extended Transactional Client Setup window.

Uninstalling the extended transactional function using Add/Remove Programs: Use the following procedure:

1. From the Windows task bar, 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 Extended Transactional Client** to select it.
4. For Windows 2000 or Windows XP:
   a. Click **Remove**. A window containing a confirmation prompt opens.
   b. Click **Yes**. The uninstall program begins and runs to completion.

   For Windows NT®:
   a. Click **Add/Remove**. The Welcome window opens.
   b. Follow the procedure in "Uninstalling the extended transactional function using the installation process" from step 2 to the end.

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 Extended Transactional Clients CD-ROM into the CD-ROM drive and enter one of the following commands at a command prompt:

- ```
  msiexec /i
  "X:\Windows\MSI\IBM WebSphere MQ Extended Transactional Client.msi"
  REMOVE="All"
  ```

  This command allows you to uninstall interactively.
  ```

- `msiexec /i "X:\Windows\MSI\IBM WebSphere MQ Extended Transactional Client.msi" /q REMOVE="All"`

  This command uninstalls in unattended mode.
- `msiexec /x "X:\Windows\MSI\IBM WebSphere MQ Extended Transactional Client.msi"`

  This command provides a confirmation prompt, and displays only a progress bar while uninstalling.
- `msiexec /x "X:\Windows\MSI\IBM WebSphere MQ Extended Transactional Client.msi" /q`

  This command uninstalls in unattended mode.

In these commands, *X* is the drive letter of your CD-ROM drive.

Chapter 4. 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 Chapter 8, "Using channels," on page 95.

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"
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

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

Transmission protocol	WebSphere MQ client	WebSphere MQ server
LU 6.2	UNIX systems[1] Windows	i5/OS UNIX systems[1] Windows z/OS
NetBIOS	Windows	Windows
SPX	Windows	Windows

Notes:

1. Except Linux (POWER platform)

Defining a connection is described in the following sections:

- "Defining a TCP/IP connection" on page 69
- "Defining an LU 6.2 connection" on page 70
- "Defining a NetBIOS connection" on page 70
- "Defining an SPX connection" on page 70

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_Q_MGR_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 client 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. See the section *Defining an SPX connection* for Windows in *WebSphere MQ Intercommunication*.

Chapter 5. 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®, Encina®, 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"
- "Microsoft Transaction Server" on page 79
- "WebSphere Application Server" on page 80

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 Chapter 3, "Installing client components," on page 21. Using the information in this section, you can then configure the extended transactional function for an XA compliant transaction manager such as CICS, Encina, 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" on page 73.

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 76.

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 77
- "Configuring for Encina" on page 78
- "Configuring for Tuxedo" on page 79

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, ENCINA, 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 75.

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)
	HP-UX	/opt/encina/lib/libEncServer.sl
	Solaris	/opt/encina/lib/libEncServer.so
	Windows systems	libEncServer
Encina	AIX	/usr/lpp/encina/lib/libEncServer.a(EncServer_shr.o)
	HP-UX	/opt/encina/lib/libEncServer.sl
	Solaris	/opt/encina/lib/libEncServer.so
	Windows systems	libEncServer
Tuxedo	AIX	/usr/lpp/tuxedo/lib/libtux.a(libtux.so.60)
	HP-UX	/opt/tuxedo/lib/libtux.sl
	Solaris	/opt/tuxedo/lib/libtux.so.60
	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]

Notes:

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.

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]

Notes:

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 Clients* or *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 Encina

You configure an extended transactional client for use by Encina by calling the tmxa_RegisterRMI function in an Encina application.

For an example of a tmxa_RegisterRMI function call, see the Encina sample program, amqsxae0.c. In the program, two arguments of the tmxa_RegisterRMI function call, openInfo and switchP, specify the following information:

openInfo An xa_open string.

switchP A pointer to an XA switch structure. Encina supports dynamic

registration of a resource manager, and so you can use either MQRMIXASwitch or MQRMIXASwitchDynamic.

The Encina sample program is also supplied as an executable module. When the module runs, it prompts the user to enter a queue manager name and an xa_open string. On AIX, HP-UX, and Solaris, the name of the module is amqsxaec and, on Windows systems, the name of the module is amqsxaec.exe.

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

To configure an extended transactional client for Microsoft Transaction Server (MTS), you must configure the WebSphere MQ base client as described in *WebSphere MQ Clients*. After you have done this, 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]

Notes:

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.

Chapter 6. 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 UNIX and Windows systems"
2. "Setting up the WebSphere MQ client" on page 86
3. "Putting a message on the queue" on page 87
4. "Getting the message from the queue" on page 87
5. "Ending verification" on page 87

The installation used for the example

These instructions assume that:

- The full WebSphere MQ product has been installed on a server:
  - The Base Product and the Client Attachment feature on z/OS
  - The Base Product and Server on other platforms
- 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 Chapter 4, "Configuring communication links," on page 67.

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 Chapter 7, "Setting up WebSphere MQ client security," on page 91 for details if you are concerned with WebSphere MQ client security issues.

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."

Create a directory to hold working files, for example `mqverify`, and make this the current directory. Then follow the steps below to set up the server workstation.

1. Create a default queue manager (called *queue.manager.1*) by entering the following command at the command prompt:

   ```
   crtmqm -q queue.manager.1
   ```

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

   ```
   strmqm
   ```

3. Start WebSphere MQ Script (MQSC) Commands by entering the following command:

   ```
   runmqsc
   ```

   MQSC does not provide a prompt, but responds with the message:

   ```
   Starting MQSC for queue manager qmname.
   ```

   where *qmname* is the name of the queue manager.

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

   ```
   DEFINE QLOCAL(QUEUE1)
   ```

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

   ```
   DEFINE CHANNEL(CHANNEL1) CHLTYPE(SVRCONN) TRPTYPE(TCP) MCAUSER(' ')
   ```

   See "Access control" on page 92 for information about `MCAUSER`.

6. Stop MQSC by typing `end` and then Enter.
7. Configure the system to start channels

   On Windows

   Start a listener by entering the following command at the command prompt:

   ```
   RUNMQLSR -t tcp -m queue.manager.1
   ```

   On UNIX systems

   Use one of the following methods:

   - Start a listener by entering the following command:

     ```
     RUNMQLSR -t tcp -m queue.manager.1
     ```

   - Or, configure the **inetd** daemon to start the MQI channels. See *WebSphere MQ Intercommunication* for details of how to do this.

Setting up a Windows server using WebSphere MQ Explorer

Create a directory to hold working files, for example `mqverify`, and make this the current directory. Then follow the steps below to set up the server connection.

- Create a default queue manager
  1. Open WebSphere MQ Explorer from the Start Menu.
  2. Right-click the folder called Queue Managers, and select New.
  3. In the first entry field, type the queue manager name, *queue.manager.1*.
  4. Make this the default queue manager.
  5. Click Next twice.
  6. Create the server connection.

7. Click Next.
8. Enter a port number of 1414 and click finish.

- Create a local queue
  1. Expand the queue manager you have just created and right-click queues.
  2. Select new local queue.
  3. Enter the queue name and click Finish.
- Define the server-connection channel
  1. Select Advanced, and right-click Channels.
  2. Select New Server-connection channel.
  3. Enter the channel name, CHANNEL1, and 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, and clicking the Show System Objects button. The listener will be running.

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-number)
   ```

Setting up the WebSphere MQ client

When an 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"

Before starting, type `ping server-address` (where `server-address` is the TCP/IP hostname 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 hostname.

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 Chapter 10, "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 hostname 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 registry for Windows. If no value is specified in the QM.INI file, or the registry for Windows, 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**

   Select a name for the client-connection, and click Next
3. Enter the following as the connection name:

   `server-connection(port)`

   Where:
   `server-address` is the TCP/IP hostname of the server
   `(port)` is the TCP/IP port number the server is listening on

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 140.

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 `/samp/bin` directory on UNIX and the `Samples\Bin` directory in Windows. Then enter the following command:

   ```
   amqsputc QUEUE1 qmgr
   ```

   where `qmgr` 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 qname 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 qmgr
   ```

   Where `qmgr` 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
```

Part 3. System administration

Chapter 7. Setting up WebSphere MQ client security 91
Authentication 91
User IDs 92
Access control 92

Chapter 8. Using channels 95
What is a channel? 95
Message Channels 95
MQI Channels 95
Defining MQI channels 96
Automatically defined channels 96
User defined channels 97
Creating one definition on the WebSphere MQ client and the other on the server 97
On the server 97
Creating a queue manager and starting MQSC on the server 97
Defining the server-connection channel . . . 97
On the WebSphere MQ client 98
Using MQSERVER 98
Using the MQCNO structure on an MQCONNX call 99
Creating both definitions on the server 99
Defining the server-connection channel 100
Defining the client-connection channel 100
Accessing client-connection channel definitions 101
Client channel definition table 101
Migrating to a later release level of WebSphere MQ. 102
Channel exits 103
Path to exits 103
Connecting a client to a queue-sharing group . . 103
Connecting to a specific queue manager . . . 104
Connecting to the generic interface 104
Creating channel definitions 104
Stopping channels 104

Chapter 9. The Secure Sockets Layer (SSL) on WebSphere MQ clients 107
Specifying that an MQI channel uses SSL 107
Specifying the location of LDAP servers that hold certificate revocation lists (CRLs) 108
When a WebSphere MQ client application issues an MQCONNX call 108
Using a client channel definition table 108
Using Active Directory on Windows. 109
Renegotiating the secret key 109
Refreshing a client's view of the SSL key repository contents and SSL settings 110
Specifying that only FIPS-certified cryptography will be used 110

Chapter 10. Using WebSphere MQ environment variables 111
MQCCSID 112
MQCHLLIB 112
MQCHLTAB. 113
Using MQCHLLIB and MQCHLTAB 113
MQIPADDRV 113
MQNAME 113
MQSERVER 114
TCP/IP default port 114
SPX default socket. 114
Using MQSERVER. 115
Example of using MQSERVER. 115
Canceling MQSERVER 115
MQSSLCRYP 115
MQSSLFIPS 116
MQSSLKEYR 116
MQSSLRESET 116

Chapter 7. 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 Version 6.0 provides Secure Sockets Layer (SSL) support for WebSphere MQ clients on the following platforms:

- AIX
- HP-UX
- Linux (x86 platform)
- Linux (zSeries platform)
- Linux (POWER platform)
- Solaris
- 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.

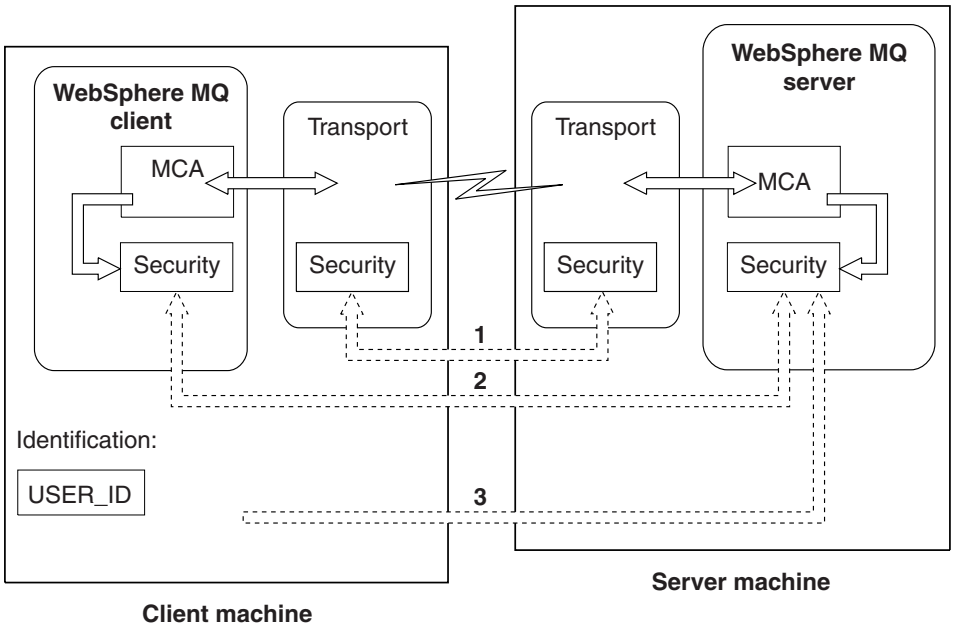

Figure 3. Security in a client-server connection

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 the user identifier associated with the process making MQI calls. For WebSphere MQ clients, the process that issues the MQI calls is the server-connection MCA. The user identifiers used by the server-connection MCA are that contained in the `MCAUserIdentifier` and `LongMCAUserIdentifier` fields of the MQCD. The contents of these fields are determined by:

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

Depending upon the combination of settings of the above, the user-identifier fields are set to appropriate values. If a server-connection security exit is provided, the user-identifier fields can be set by the exit. Otherwise they are determined as follows:

- If the server-connection channel MCAUSER attribute is nonblank, this value is used.
- If the server-connection channel MCAUSER attribute is blank, the user ID received from the client is used.

When the user-identifier 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.

Chapter 8. Using channels

This chapter discusses the following topics:

- "What is a channel?"
- "Defining MQI channels" on page 96
- "Creating a queue manager and starting MQSC on the server" on page 97
- "Creating one definition on the WebSphere MQ client and the other on the server" on page 97
- "Creating both definitions on the server" on page 99
- "Channel exits" on page 103
- "Connecting a client to a queue-sharing group" on page 103
- "Migrating to a later release level of WebSphere MQ" on page 102

What is a channel?

A channel is a logical communication link between an WebSphere MQ client and an 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. More information on message channels can be found in the *WebSphere MQ Intercommunication*

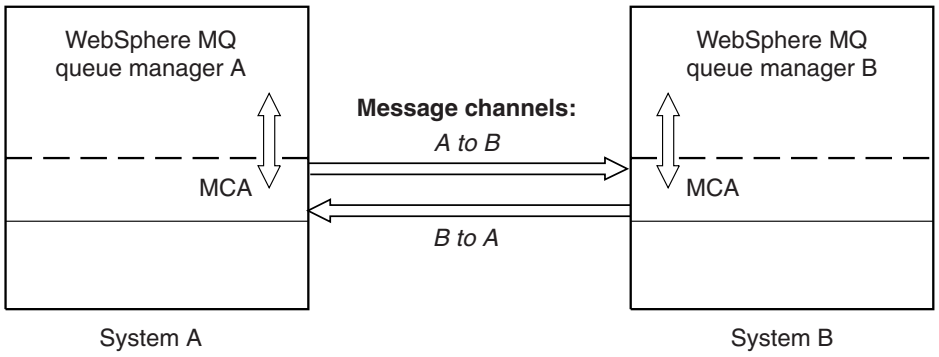

Figure 4. Message channels between two queue managers

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").

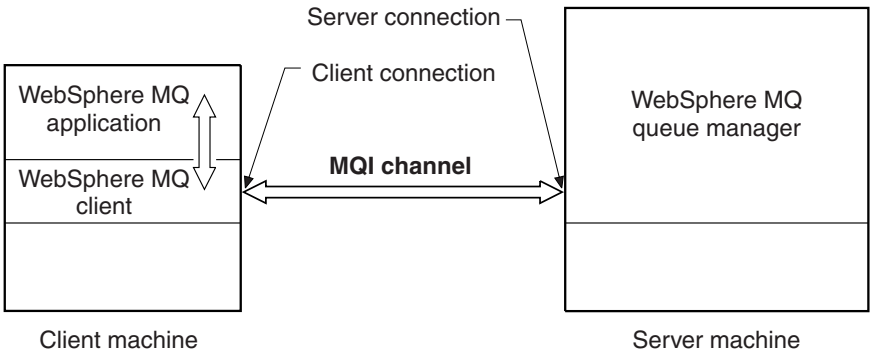

Figure 5. Client-connection and server-connection on an MQI channel

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 103).

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 an WebSphere MQ client environment, will communicate.

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 99 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=CHAN1/TCP/MCID66499
```

or, on a UNIX system:

```
export MQSERVER=CHAN1/TCP/'MCID66499'
```

Note: To change the TCP/IP port number, see "MQSERVER" on page 114.

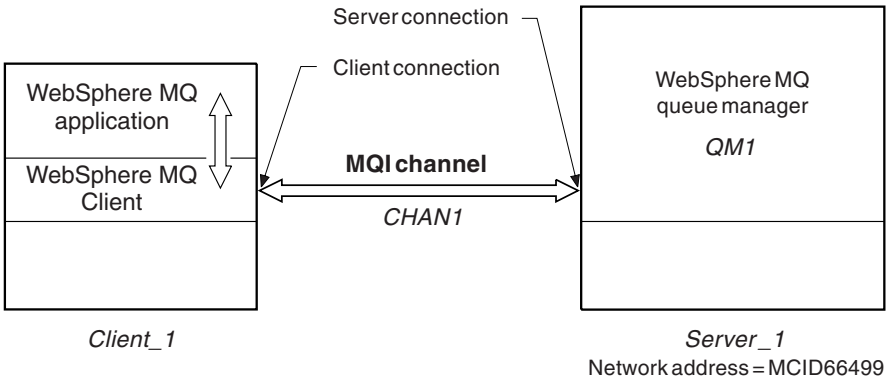

Figure 6. Simple channel definition

Some more examples of a simple channel definition on Windows are:

```
SET MQSERVER=CHAN1/TCP/9.20.4.56
SET MQSERVER=CHAN1/NETBIOS/BOX643
```

Some examples of a simple channel definition on a UNIX system are:

```
export MQSERVER=CHAN1/TCP/'9.20.4.56'
export MQSERVER=CHAN1/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 Chapter 10, "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 97
2. On the server machine, define a channel with your chosen name and a channel type of *server-connection*.

   For example:

   ```
   DEFINE CHANNEL(CHAN2) CHLTYPE(SVRCONN) TRPTYPE(TCP) +
   DESCR('Server-connection to Client_2')
   ```

   This channel definition is associated with the queue manager running on the server.

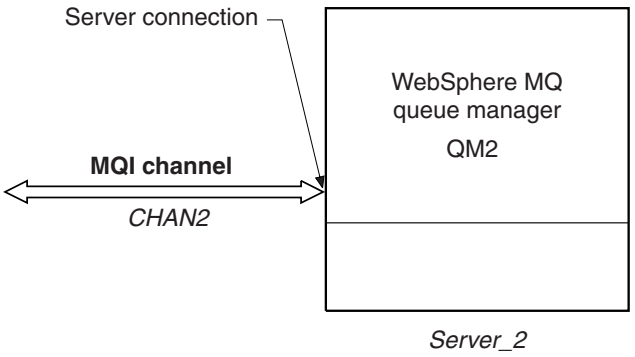

Figure 7. Defining the server-connection channel

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 Chapter 13, "Running applications on WebSphere MQ clients," on page 129.

For example:

```
DEFINE CHANNEL(CHAN2) 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" on page 101.

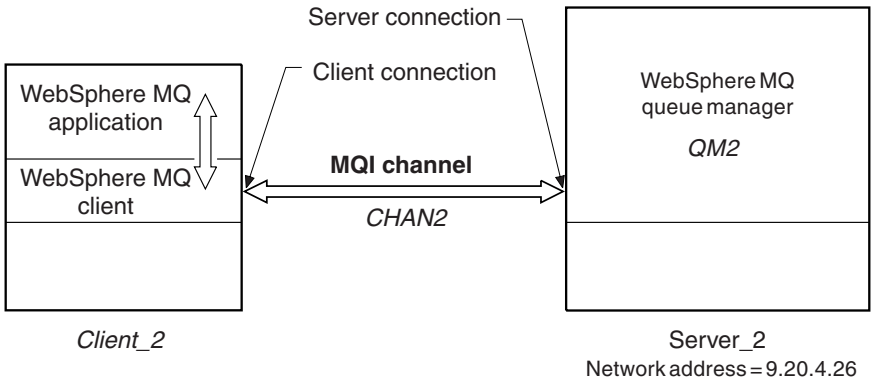

Figure 8. Defining the client-connection channel

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 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

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.

Migrating to a later release level of WebSphere MQ

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.

For example, a Version 6.0 WebSphere MQ client can use a client channel definition table that has been prepared by a Version 5.3 queue manager but a Version 5.3 client cannot use a client channel definition table that has been prepared by a Version 6.0 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 `mqs.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 `mqs.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.

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(CHAN1) 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(CHAN1) 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.

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).

Chapter 9. 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 108
- "Renegotiating the secret key" on page 109
- "Refreshing a client's view of the SSL key repository contents and SSL settings" on page 110
- "Specifying that only FIPS-certified cryptography will be used" on page 110

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"
3. "Using Active Directory on Windows" on page 109

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 secret key. You can make a client renegotiate the key by using the KeyResetCount field in the MQSCO structure on an MQCONNX call or by using the environment variable MQSSLRESET. Both 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 secret key is renegotiated. Specifying a value of 0 for MQSSLRESET or for KeyResetCount indicates that secret keys are never

renegotiated. For an MQCONNX call, if both MQSSLRESET and KeyResetCount are specified, the value of the latter is used.

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 116.

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, or the cryptographic hardware parameters, all the SSL connections must be ended. Once they are all ended they can be restarted and will reflect the changes made in the client-connection channels that the application is using to connect to the queue manager. 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

To specify that an SSL channel must only use FIPS-certified cryptography, set either the environment variable MQSSLFIPS or the FipsRequired field in the MQSCO structure to MQSSL_FIPS_YES (the default value is MQSSL_FIPS_NO). 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.

This is affected by the use of cryptographic hardware. See "MQSSLFIPS" on page 116.

If the MQSSLFIPS and FipsRequired variables are both set but with inconsistent values, the FipsRequired value takes precedence.

If this variable is set to MQSSL_FIPS_YES but the platform is not a FIPS-certified platform, all SSL connections fail with MQRC_SSL_INITIALIZATION_ERROR.

If this variable is set to MQSSL_FIPS_YES and a non-FIPS CipherSpec is specified for a connection, the connection fails with MQRC_SSL_INITIALIZATION_ERROR.

Chapter 10. 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 113
- "MQSERVER" on page 114
- "MQSSLCRYP" on page 115
- "MQSSLFIPS" on page 116
- "MQSSLKEYR" on page 116
- "MQSSLRESET" on page 116

Commands are available on all the WebSphere MQ client platforms unless otherwise stated.

Notes:

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 101 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 119 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
```

Notes:

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 101.

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

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 70 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 is used to define a minimal channel. It cannot be used to define an SSL channel, or a channel with channel exits. It specifies the location of the WebSphere MQ server and the communication method to be used. Note that *ConnectionName* must be a fully-qualified network name. The *ChannelName* cannot contain the forward slash (/) character because it 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 4 MB is used, so larger messages cannot flow across this channel. For larger messages a client-connection channel must be defined using DEFINE CHANNEL, on the server, with MAXMSGL set to a larger figure, or using an MQCONNX call with MaxMsgLength set in the MQCD referred to in the MQCNO structure.

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 `qm.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 69.

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=CHAN1/TCP/'9.20.4.56(2002)'
export MQSERVER=CHAN1/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 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 as described in the *WebSphere MQ Script (MQSC) Command Reference*.

To set this variable use one of these commands:

- On Windows systems:

  ```
  SET MQSSLCRYP=string
  ```

- On UNIX systems:

  ```
  export MQSSLCRYP=string
  ```

MQSSLFIPS

This specifies if 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 may, or may not, be FIPS-certified to a particular level. This depends on the hardware product in use. The values are the same as for the SSLFIPS parameter of the ALTER QMGR command as described in the *WebSphere MQ Script (MQSC) Command Reference*.

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 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 filename without an extension. For full details, see the SSLFIPS parameter of the ALTER QMGR command as described in the *WebSphere MQ Script (MQSC) Command Reference*.

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 secret key is renegotiated. See "Renegotiating the secret key" on page 109 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 secret keys are never renegotiated.

To set this variable use one of these commands:

- On Windows systems:

  ```
  SET MQSSLRESET=integer
  ```

- On UNIX systems:

  ```
  export MQSSLRESET=integer
  ```

Part 4. Application programming

Chapter 11. Using the message queue interface (MQI) 119
Limiting the size of a message 119
Choosing client or server coded character set identifier (CCSID) 119
CCSID and encoding fields - multiple puts . . 120
Designing applications 120
Using MQINQ 120
Using syncpoint coordination 120
Using MQCONNX 121
Shared connection handles on MQCONNX . . 121

Chapter 12. Building applications for WebSphere MQ clients 123
Running applications in the WebSphere MQ client environment. 123
Triggering in the client environment. 124
Process definition 124
Trigger monitor 124
CICS applications (non-z/OS) 125
Linking C applications with the WebSphere MQ client code 125
Linking C++ applications with the WebSphere MQ client code 126
Linking COBOL applications with the WebSphere MQ client code 126
Linking Visual Basic applications with the WebSphere MQ client code 126

Chapter 13. Running applications on WebSphere MQ clients 129
Using environment variables 129
Using the MQCNO structure 130
Using DEFINE CHANNEL 130
Role of the client channel definition table 130
Multiple queue managers 130
Queue-sharing groups 130
Examples of MQCONN calls 131
What the examples demonstrate 132
Example 1. Queue manager name includes an asterisk (*) 132
Example 2. Queue manager name specified . . 133
Example 3. Queue manager name is blank or an asterisk (*) 133

Chapter 14. Preparing and running extended transactional client applications 135
Preparing and running CICS, Encina, and Tuxedo applications 135
Sample programs 135
Error log messages 137
Preparing and running Microsoft Transaction Server applications 137
Preparing and running WebSphere MQ JMS applications 137

Chapter 15. Solving problems 139
WebSphere MQ client fails to make a connection 139
Stopping WebSphere MQ clients 139
Error messages with WebSphere MQ clients . . . 140
Using trace on Windows 140
File names for trace files. 140
How to examine First Failure Support Technology (FFST) files 140
Using trace on AIX 141
Using trace on HP-UX, Solaris and Linux 141
File names for trace files. 142
How to examine FFSTs 142

Chapter 11. 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)"
- "Designing applications" on page 120
- "Using MQINQ" on page 120
- "Using syncpoint coordination" on page 120
- "Using MQCONNX" on page 121

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.

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, this is done by the client support code.

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 an 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 5 for details.

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 99. 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*.

Chapter 12. Building applications for WebSphere MQ clients

This chapter lists points to consider when running an application in an 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 124
- "Linking C applications with the WebSphere MQ client code" on page 125
- "Linking C++ applications with the WebSphere MQ client code" on page 126
- "Linking COBOL applications with the WebSphere MQ client code" on page 126
- "Linking Visual Basic applications with the WebSphere MQ client code" on page 126

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	✓	✓	✓	
HP-UX	✓	✓	✓	
Linux	✓	✓		
Solaris	✓	✓	✓	
Windows	✓	✓	✓	✓

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:

- It 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
- It does not need to use any of the exceptions listed in "What applications run on a WebSphere MQ client?" on page 6

Note: The libraries 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. It will have one connection to each server it requires, a connection being established with each **MQCONN** or **MQCONNX** call the application issues.
- An application sends and gets messages synchronously.
- All data conversion is done by the server, but see also "MQCCSID" on page 112.

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 the server. 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

Chapter 13. Running applications on WebSphere MQ clients

This chapter 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"
- "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 MQCONN calls" on page 131

When an application running in an 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 "Migrating to a later release level of WebSphere MQ" on page 102.

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 99.

Using DEFINE CHANNEL

If you use the MQSC DEFINE CHANNEL command, the details you provide are placed in the client channel definition table. This file is accessed by the client, in alphabetic channel name sequence, to determine the channel an application will use.

The contents of the *QMgrName* parameter of the MQCONN or MQCONNX call determines which queue manager the client connects to.

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 Chapter 10, "Using WebSphere MQ environment variables," on page 111 for information about environment variables.

Multiple queue managers

You might choose to define connections to more than one server machine because:

- You need a backup system.
- You want to be able to move your queue managers without changing any application code.
- You need to access multiple queue managers, and this requires the least resource.

Define your client-connection and server-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.

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.

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 104. 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 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.

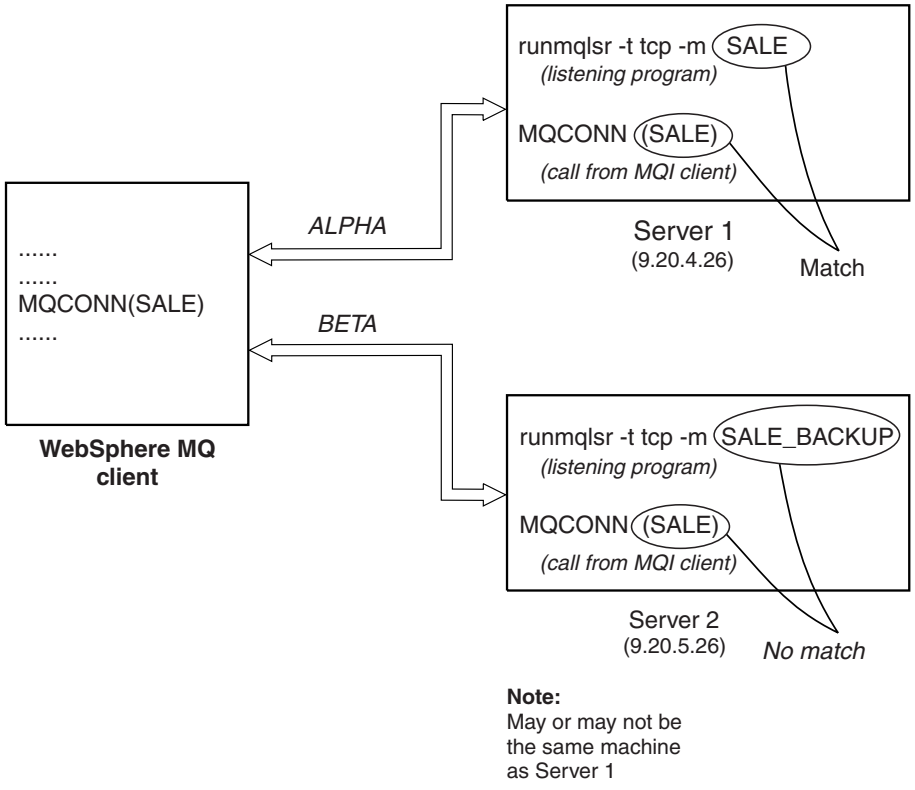

Figure 9. MQCONN example

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:

```
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. 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 132.

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.

Chapter 14. Preparing and running extended transactional client applications

This chapter contains the following sections:

- "Preparing and running CICS, Encina, and Tuxedo applications"
- "Preparing and running Microsoft Transaction Server applications" on page 137
- "Preparing and running WebSphere MQ JMS applications" on page 137

Preparing and running CICS, Encina, and Tuxedo applications

To prepare CICS, Encina, 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, Encina, 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 mqic32.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 on page 136 lists the CICS, Encina, and Tuxedo sample programs that are supplied for use on AIX, HP-UX, and Solaris client systems. Table 23 on page 136 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	-
Encina program	amqsxae0.c	amqsxaec
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	-
Encina transaction	amqsxae0.c	amqsxaec
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, Encina, 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 mqic32xa.lib instead of mqic32.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

Chapter 15. Solving problems

This chapter discusses the following topics:

- "WebSphere MQ client fails to make a connection"
- "Stopping WebSphere MQ clients"
- "Error messages with WebSphere MQ clients" on page 140
- "Using trace on Windows" on page 140
- "Using trace on AIX" on page 141
- "Using trace on HP-UX, Solaris and Linux" on page 141

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 an 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.

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

A client on Windows uses the following commands for the client trace facility:

strmqtrc
: to start tracing

endmqtrc
: to end tracing

File names for trace files

The output file is created in the *mqmtop*\trace directory.

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.

Notes:

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.

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.

The WebSphere MQ trace utility is explained in detail in the *WebSphere MQ System Administration Guide* manual.

Using trace on AIX

WebSphere MQ for AIX use the standard AIX system trace. Tracing is a two step process:

1. Gather the data
2. Format the results

WebSphere MQ uses two trace hook identifiers:

X'30D' This event is recorded by WebSphere MQ on entry to or exit from a subroutine.

X'30E' This event is recorded by WebSphere MQ to trace data such as that being sent or received across a communications network.

Trace provides detailed execution tracing to help you to analyze problems. IBM service support personnel might ask for a problem to be recreated with trace enabled. The files produced by trace can be very large, so it is important to qualify a trace, where possible. For example, you can optionally qualify a trace by time and by component.

The best way to trace a single WebSphere MQ application is to run trace synchronously, and run the application from within the trace program. For example:

```
trace -j30D,30E -o trace.trc
```

At the prompt, enter the command that you want traced, prefixed by an exclamation mark (!). For example, type:

```
!amqsputc TESTQ
```

When the command is complete, type `quit` to stop trace and exit the trace program.

To view the trace, you must format the file using the `trcrpt` command. You can then view the trace. The following example shows you how to produce the file `trace.fmt`, which you can then view:

```
trcrpt -t /usr/mqm/lib/amqtrc.fmt trace.trc > trace.fmt
```

If you want to trace more than one application at the same time, or you want to start trace while an application is already running, start trace asynchronously using the `-a` flag. For example:

```
trace -a -j30D,j30E -o trace.trc
```

Trace continues to run until you stop it using the `trcstop` command. The trace file can then be formatted as before using the `trcrpt` command.

The WebSphere MQ for AIX trace utility is explained in detail in the *WebSphere MQ System Administration Guide* manual.

Using trace on HP-UX, Solaris and Linux

HP-UX and Solaris use the following commands for the WebSphere MQ client trace facility:

strmqtrc -e
to start early tracing
endmqtrc -e
to end early tracing
dspmqtrc <filename>
to display a formatted trace file

For more information about the trace commands, see the *WebSphere MQ System Administration Guide* manual.

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.TRC
```

where ppppp is the process ID (PID) of the process producing the trace.

Notes:

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.

How to examine FFSTs

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.

The WebSphere MQ trace utility is explained in detail in the *WebSphere MQ System Administration Guide*.

Appendix A. 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.

Appendix B. Notices

This information was developed for products and services offered in the United States. IBM may not offer the products, services, or features discussed in this information in other countries. Consult your local IBM representative for information on the products and services currently available in your area. Any reference to an IBM product, program, or service is not intended to state or imply that only that IBM product, program, or service may be used. Any functionally equivalent product, program, or service that does not infringe any IBM intellectual property right may be used instead. However, it is the user's responsibility to evaluate and verify the operation of any non-IBM product, program, or service.

IBM may have patents or pending patent applications covering subject matter described in this information. The furnishing of this information does not give you any license to these patents. You can send license inquiries, in writing, to:

IBM Director of Licensing
IBM Corporation
North Castle Drive
Armonk, NY 10504-1785
U.S.A.

For license inquiries regarding double-byte (DBCS) information, contact the IBM Intellectual Property Department in your country or send inquiries, in writing, to:

IBM World Trade Asia Corporation
Licensing
2-31 Roppongi 3-chome, Minato-ku
Tokyo 106, Japan

The following paragraph does not apply to the United Kingdom or any other country where such provisions are inconsistent with local law:
INTERNATIONAL BUSINESS MACHINES CORPORATION PROVIDES THIS PUBLICATION "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY, OR FITNESS FOR A PARTICULAR PURPOSE. Some states do not allow disclaimer of express or implied warranties in certain transactions, therefore this statement may not apply to you.

This information could include technical inaccuracies or typographical errors. Changes are periodically made to the information herein; these changes will be incorporated in new editions of the information. IBM may make improvements and/or changes in the product(s) and/or the program(s) described in this information at any time without notice.

Any references in this information to non-IBM Web sites are provided for convenience only and do not in any manner serve as an endorsement of those Web sites. The materials at those Web sites are not part of the materials for this IBM product and use of those Web sites is at your own risk.

IBM may use or distribute any of the information you supply in any way it believes appropriate without incurring any obligation to you.

Licensees of this program who wish to have information about it for the purpose of enabling: (i) the exchange of information between independently created programs and other programs (including this one) and (ii) the mutual use of the information which has been exchanged, should contact:

IBM United Kingdom Laboratories,
Mail Point 151,
Hursley Park,
Winchester,
Hampshire,
England
SO21 2JN.

Such information may be available, subject to appropriate terms and conditions, including in some cases, payment of a fee.

The licensed program described in this information and all licensed material available for it are provided by IBM under terms of the IBM Customer Agreement, IBM International Programming License Agreement, or any equivalent agreement between us.

Information concerning non-IBM products was obtained from the suppliers of those products, their published announcements or other publicly available sources. IBM has not tested those products and cannot confirm the accuracy of performance, compatibility or any other claims related to non-IBM products. Questions on the capabilities of non-IBM products should be addressed to the suppliers of those products.

Trademarks

The following terms are trademarks of the IBM Corporation in the United States, or other countries, or both:

AIX	CICS	DB2
Encina	First Failure Support Technology	FFST
IBM	IBMLink	IMS
i5/OS	iSeries	MQSeries
POWER	TXSeries	VisualAge
VTAM	WebSphere	z/OS
zSeries		

Intel is a registered trademark of Intel Corporation in the United States and/or other countries.

Microsoft, Windows, and Windows NT are trademarks of Microsoft Corporation in the United States, other countries, or both.

Java and all Java-based trademarks and logos are trademarks of Sun Microsystems, Inc. in the United States, other countries, or both.

Linux is a trademark of Linus Torvalds in the United States, other countries, or both.

UNIX is a registered trademark of The Open Group in the United States and other countries.

Other company, product, or service names may be trademarks or service marks of others.

Index

A

access control 92
Active Directory
  accessing client-connection channel definitions 101
  locating LDAP servers that hold CRLs 109
  specifying that an MQI channel uses SSL 107
advanced installation
  client 53
advantages of using WebSphere MQ clients 6
AIX
  uninstalling the extended transactional client 30
AIX client
  components for 24
  custom installation 26
  hardware and software 15
  installing
    without SSL support 27
  migrating to and from SSL support 30
  removing 30
  trace 141
AMQCLCHL.TAB 101
AMQSGETC sample program 83, 87
AMQSGETW sample program 83
amqsputc sample program 87
AMQSPUTC sample program 83
AMQSPUTW sample program 83
applications
  building 123
  connected to multiple queue managers 129
  connected to multiple servers 129
  connection to server 129
  in different environments 123
  non-version 6.0 13
  version 6.0 13
applications on WebSphere MQ clients 6
ApplType 124
authentication 91
authentication information object 108
automatic definition of channels 96

B

benefits of using WebSphere MQ clients 6
building applications 123
building applications for WebSphere MQ clients
  MQI 103

C

C applications
  linking 125
C++ applications
  linking 123, 126
  using 123
CCSID (coded character set identifier) 119
CEDA 125
certificate revocation list (CRL) 108
channel
  client connection 129
  client-connection 96
  definition
    automatic 96
    by user 97
    connecting to a queue-sharing group 104
    on the client and server 97
    using MQSC commands 97
  message 95
  MQI 95
  overview 95
  server-connection 96, 100
  starting 84
  user-defined 97
channel definition
  maximum message length 119
  on the client 98
  on the server 97, 99
  overview 95
channel exits
  ClientExitPath 103
  ExitsDefaultPath 103
  overview 103
  path to exits 103
  receive 103
  security 103
  send 103
channel initiator, starting
  z/OS example 86
channels
  stopping 104
CICS
  configuring an extended transactional client 77
  preparing and running client applications 135
  sample client programs 135
CICS applications (non-z/OS)
  CEDA 125
  CICSENV.CMD file 125
  environment specifics 125
  MQSERVER 125
client
  installation
    advanced methods 53
    LAN 50
    unattended 51
  unattended uninstall 62
  uninstalling 61
client applications
  CICS, Encina, and Tuxedo sample programs 135
client applications *(continued)*
  preparing and running
    CICS, Encina, and Tuxedo 135
    Microsoft Transaction Server 137
    WebSphere Application Server 137
    WebSphere MQ JMS 137
Client CD 21
client channel definition table
  directory path 112
  how it is used 130
  locating LDAP servers that hold CRLs 108
  migrating to a later release level of WebSphere MQ 102
  name of 113
  specifying that an MQI channel uses SSL 107
  where to find it 101, 129
client library file 125
client setup
  example 86
client to server connection 4, 96
client-connection channel
  defining 96
  example 86
client-connection, defining 100
ClientExitPath 103
clients
  removing 23
clients overview 3
CLNTCONN 100
COBOL
  link libraries 126
  linking applications 126
coded character set identifier (CCSID) 119
commands
  MQSC 97
communication protocol
  LU 6.2 67
  NetBIOS 67
  overview 13
  SPX 67
  TCP/IP 67
communication type 67
communications
  configuring 67
  overview 13
compact installation
  client 48
components
  Linux client 36
  Solaris client 42
configuring an extended transactional client
  CICS 77
  Encina 78
  Microsoft Transaction Server 79
  Tuxedo 79
  WebSphere Application Server 80

configuring an extended transactional client *(continued)*
WebSphere MQ JMS 80
XA compliant transaction managers 71
configuring communications 67
connection
client to server 4, 96
LU 6.2 70
NetBIOS 70
overview 13
queue managers 129
server to client 96
SPX 70
TCP/IP 69
to queue-sharing group 103
creating
groups
on AIX client 24
on Linux client 38
user ID
on AIX client 24
on Linux client 38
CRL
See certificate revocation list (CRL)
CRL information 108
CSQUTIL
See WebSphere MQ utility program (CSQUTIL)
custom installation
AIX client 26
client 48
customization
z/OS example 85

D

data compression 103
data conversion 123
data encryption 103
defining channels
connecting to a queue-sharing group 104
on the client 98
on the client and server 97
on the server 97, 99
overview 95
using MQSC 130
using MQSC commands 97
delete queue manager 87
dltmqm 87
DQM, starting
z/OS example 86
dspmqtrc trace command 141
dynamic registration 77

E

Encina
configuring an extended transactional client 78
preparing and running client applications 135
sample client programs 135
ending verification 87
endmqm 87
endmqtrc trace command 140, 141
environment variables 111
change setting 111
connecting to a client 129
creating channel definitions 96
display current setting 111
MQ_USER_ID 92
MQCCSID 112
MQCHLLIB 112, 129
MQCHLTAB 113, 129
MQIPADDRV 113
MQNAME 113
MQSERVER 114, 129
MQSSLCRYP 115
MQSSLFIPS 116
MQSSLKEYR 116
MQSSLRESET 116
error log
overview 139
error messages 139
example
customize z/OS 85
inetd setup 84
installation verification 83
local queue, creating 84
MQSC, starting 84
MQSC, stopping 84
queue manager
creating 84
starting 84
server-connection channel, creating 84
setting up the server 83
exits
overview 103
receive 103
security 92, 103
send 103
ExitsDefaultPath 103
extended transactional client
configuring
CICS 77
Encina 78
Microsoft Transaction Server 79
Tuxedo 79
WebSphere Application Server 80
WebSphere MQ JMS 80
XA compliant transaction managers 71
defined 5
hardware and software requirements 12
installing
Solaris, basic method 43
Solaris, unattended 44
Windows, basic method 49
Windows, from a LAN server 50
Windows, unattended 53
introduction 5
supported platforms 11
supported server platforms 12
supported transaction managers 11
uninstalling
AIX 30
HP-UX 34
Linux 41
Solaris 45
extended transactional client *(continued)*
uninstalling *(continued)*
Windows, from a command prompt 64
Windows, introduction 63
Windows, using Add/Remove Programs 64
Windows, using the installation process 64

F

features
for a client installation 46
unattended installation 53
FFST, examining 140, 142
First Failure Support Technology 140

G

getting a message from a queue
example 87
groups, creating
on AIX client 24
on HP-UX client 32
on Linux client 38
on Solaris client 42

H

hardware and software requirements 12
hardware requirements
AIX client 15
HP-UX client 16
Linux client 17
Solaris client 19
Windows client 20
HP-UX
uninstalling the extended transactional client 34
HP-UX client
groups, creating 32
hardware and software 16
installing 33
migrating to and from SSL support 34
national language support 34
trace 141
uninstalling 34
user ID, creating 32

I

i5/OS client 4
inetd setup 84
z/OS example 86
installation
advanced methods
MQParms command 59
transforms 57
client
advanced methods 53
LAN 50
Msiexec 53
unattended 51

installation *(continued)*
client types 46
clients 21
compact
client 48
custom
client 48
modifying
using Add/Remove Programs 49
using the Client CD 48
preparing for 11
response file
client 52
server 21
typical
client 46
verification 83
installing 21
AIX client
without SSL support 27
AIX client (custom install) 26
client on the server 22
HP-UX client 33
Linux client 36
Solaris client 43
Windows client 46
installing an extended transactional client
Solaris
basic method 43
unattended 44
Windows
basic method 49
from a LAN server 50
unattended 53

L

LAN installation
client 50
LAN installation on Windows 50
language
installation from client CD 47
unattended installation 57
languages supported 123
LDAP server
See Lightweight Directory Access Protocol (LDAP) server
library file, client 125
Lightweight Directory Access Protocol (LDAP) server 108
linking applications for WebSphere MQ clients
MQI 103
linking with WebSphere MQ client code
C applications 125
C++ applications 126
COBOL applications 126
Visual Basic applications 126
Linux
uninstalling the extended transactional client 41
Linux client
components 36
hardware and software 17
installing 36
migrating to and from SSL support 41
Linux client *(continued)*
national language support 40
uninstalling 40
listener 67
listener, starting
z/OS example 86
listening on LU 6.2 70
listening on TCP/IP 69
local queue
example 84
local queue, creating
example (z/OS) 85
LU 6.2 67
LU 6.2 connections 70
on a client 70
on a server 70

M

MCAUSER 92
MCAUserIdentifier 92
message
errors on UNIX systems 140
errors on Windows 140
maximum length 119
message channel 95
Message Queue Interface (MQI) 119
messages
translated 30
Microsoft Transaction Server (MTS)
configuring an extended transactional client 79
preparing and running client applications 137
migrating to and from SSL support
AIX client 30
HP-UX client 34
Linux client 41
Solaris client 45
modifying the installation
using Add/Remove Programs
client 49
using the Client CD 48
MQ_USER_ID 92
MQAIR structure 108
MQBACK 120
MQCCSID
what it does 112
MQCD structure
introduction 99
specifying that an MQI channel uses SSL 107
MQCHLLIB
how it is used 129
introduction 101
what is does 112
MQCHLTAB
how it is used 129
introduction 101
what it does 113
MQCMIT 120
MQCNO structure
introduction 99
locating LDAP servers that hold CRLs 108
specifying that an MQI channel uses SSL 107
MQCONN 129
MQCONN or MQCONNX failure 139
MQCONNX 129
MQCONNX call
defining a client-connection channel 99
locating LDAP servers that hold CRLs 108
specifying that an MQI channel uses SSL 107
Using 121
MQI
application in client environment 123
building applications for WebSphere MQ clients 103
linking applications for WebSphere MQ clients 103
MQI (Message Queue Interface) 119
MQI channel 3
MQI channels 95
MQINQ 120
MQIPADDRV
what it does 113
MQNAME 113
MQParms command 59
MQParms.ini 59
mqs.ini file
path to exits 103
role in connecting a client 129
MQSC
commands 97
starting 97
example 84
stopping 84
MQSCO structure
introduction 99
locating LDAP servers that hold CRLs 108
MQSeries server
name of 114
MQSERVER
cancelling 115
defining a client-connection channel 86
how it is used 129
relationship with client channel definition table 130
specifying that an MQI channel uses SSL 107
using 98
what it does 114
MQSSLCRYP 115
MQSSLFIPS 116
MQSSLKEYR 116
MQSSLRESET 116
Msiexec command 53
with response file 58
MTS
See Microsoft Transaction Server (MTS)
multiple queue managers 130

N

namelist 108
national language
installation from client CD 47

national language *(continued)*
unattended installation 57
national language support 30
HP-UX client 34
Linux client 40
NetBIOS 67
connections 70
NLSPATH environment variable 30
non-version 6.0 clients 13

P

parameter file
contents 60
creating 59
example 60
password 92
path to exits
mqs.ini file 103
platforms
for WebSphere MQ clients 11
for WebSphere MQ servers 11
preparing client applications
CICS 135
Encina 135
Microsoft Transaction Server 137
Tuxedo 135
WebSphere Application Server 137
WebSphere MQ JMS 137
preparing for installation 11
problem determination 139
process definition 124
putting a message on the queue
example 87

Q

queue manager
definition 84
deleting 87
maximum message length 119
starting 84
starting (z/OS example) 85
stopping 87
queue managers 129
queue-sharing groups 103, 130

R

receive exit 103
removal response file format
client 63
removing
client 61
using Add/Remove Programs
Windows client 62
removing an AIX client 30
removing an extended transactional client
AIX 30
HP-UX 34
Linux 41
Solaris 45
Windows
from a command prompt 64
introduction 63
using Add/Remove Programs 64
removing an extended transactional client *(continued)*
Windows *(continued)*
using the installation process 64
resource manager 143
response file
client 52, 58
removal 63
with Msiexec command 58
return codes 139
runmqtmc 124
running client applications
CICS 135
Encina 135
Microsoft Transaction Server 137
Tuxedo 135
WebSphere Application Server 137
WebSphere MQ JMS 137

S

sample client programs 135
sample programs
AMQSGETC 83, 87
AMQSGETW 83
amqsputc 87
AMQSPUTC 83
AMQSPUTW 83
GET 83
PUT 83
Secure Sockets Layer (SSL) 107
security
access control 92
authentication 91
exit 92
on a WebSphere MQ client 91
password 92
user ID 92
security exit 103
send exit 103
server
connecting to a client 129
connection, defining 100
platform support 11
Server CD 21
server queue manager 3
server to client connection 96
server-connection channel
defining 96
example (not z/OS) 84
example (z/OS) 85
iSeries example 85
setmqcap command 22
setmqcrl command 109
setmqscp command
introduction 101
specifying that an MQI channel uses SSL 107
setting up the server
example 83
example (Windows) 84
i5/OS example 85
using Windows explorer 84
z/OS example 85
setting up WebSphere MQ clients 6
shared queues 103, 130
silent installation
client 51
Solaris 44
Windows 53
silent uninstall
client 62
simple client-connection channel
definition 98
SMIT
installing AIX client 26
using to create IDs 25
software requirements
AIX client 15
HP-UX client 16
Linux client 17
Solaris client 19
Windows client 20
Solaris
installing the extended transactional client
basic method 43
unattended 44
uninstalling the extended transactional client 45
Solaris client
components 42
hardware and software 19
installing 43
migrating to and from SSL support 45
trace 141
translated messages 44
uninstalling 45
solving problems 139
SPX 67
connections 70
default socket 114
SSL
See also Secure Sockets Layer (SSL)
key
renegotiating 109
SSLCIPH parameter 107
SSLCipherSpec field 107
SSLCRLNameList attribute 108
stop queue manager 87
strmqtrc trace command 140, 141
supported server platforms 12
supported transaction managers 11
SVRCONN 100
switch load file 77
switch structure
See XA switch structure
syncpoint 143
considerations 120
coordination 125
syncpoint control 143
syncpoint coordinator
See transaction manager
system administration 91
System Management Interface Tool (SMIT)
installing AIX client 26
using to create IDs 25

T

task list 6
TCP/IP 67
  default port 114
TCP/IP connection
  on a client 69
  on a server 69
TCP/IP connections 69
trace
  AIX 141
  HP-UX 141
  Solaris 141
  Windows 140
trace command
  dspmqtrc 141
  endmqtrc 140, 141
  strmqtrc 140, 141
transaction
  See unit of work
transaction manager
  definition 143
  external 143
transforms
  Msiexec command 57
translated messages
  client 30
  HP-UX client 34
  Linux client 40
  Solaris client 44
transmission protocol
  LU 6.2 67
  NetBIOS 67
  SPX 67
  TCP/IP 67
trigger monitor for WebSphere MQ clients 124
Tuxedo
  configuring an extended transactional client 79
  preparing and running client applications 135
  sample client programs 135
two phase commit protocol 144
types of client installation 46
typical installation
  client 46

U

unattended installation
  client 51
  Solaris 44
  Windows 53
unattended removal
  client 62
uninstallation response file format
  client 63
uninstalling
  client 61
  HP-UX client 34
  Linux client 40
  Solaris client 45
  using Add/Remove Programs
    Windows client 62
uninstalling an extended transactional client
  AIX 30
  HP-UX 34
  Linux 41
  Solaris 45
  Windows
    from a command prompt 64
    introduction 63
    using Add/Remove Programs 64
    using the installation process 64
unit of work 143
UNIX systems clients
  error messages 140
user definition of channels 97
user ID 92
user ID, creating
  on AIX client 24
  on HP-UX client 32
  on Linux client 38
  on Solaris client 42

V

verification, ending 87
verifying installation 83
  example 83
Version 6.0 clients 13
Visual Basic
  link libraries 126
  linking 126
  linking applications 126

W

WebSphere Application Server
  configuring an extended transactional client 80
  preparing and running client applications 137
WebSphere MQ classes for Java Message Service
  See WebSphere MQ JMS
WebSphere MQ client 3
WebSphere MQ Client CD 21
WebSphere MQ environment variables 111
WebSphere MQ extended transactional client 5
  See extended transactional client
WebSphere MQ JMS
  configuring an extended transactional client 80
  preparing and running client applications 137
WebSphere MQ Script (MQSC) commands 97
WebSphere MQ server
  platform support 11
WebSphere MQ Server CD 21
WebSphere MQ utility program (CSQUTIL)
  creating a client channel definition file 102
  creating CRL information in a client channel definition file 109
Windows
  installing the extended transactional client
    basic method 49
    from a LAN server 50
    unattended 53
  uninstalling the extended transactional client
    from a command prompt 64
    introduction 63
    using Add/Remove Programs 64
    using the installation process 64
Windows client
  error messages 140
  hardware and software 20
  installing 46
  trace 140
  uninstalling using Add/Remove Programs 62

X

XA compliant transaction managers
  configuring an extended transactional client 71
XA Interface 143
XA switch structure
  introduction 72
  supplied with an extended transactional client 76
  use by CICS 77
  use by Encina 78
  use by Tuxedo 79
xa_open string
  example 74
  format 73
  how it is used 74
  introduction 72
  use by CICS 77
  use by Encina 78
  use by Tuxedo 79

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.

GC34-6590-00

Spine information:

WebSphere MQ

WebSphere MQ Clients