WebSphere MQ

Extended Transactional Clients

SC34-6275-00

Note!
Before using this information and the product it supports, be sure to read the general information under "Notices", on page 39.

First edition (February 2003)

This edition applies to the following WebSphere MQ Version 5.3 products:
- WebSphere MQ for AIX
- WebSphere MQ for HP-UX
- WebSphere MQ for iSeries
- WebSphere MQ for Linux for Intel
- WebSphere MQ for Linux for zSeries
- WebSphere MQ for Solaris
- WebSphere MQ for Windows

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

Contents

Tables

About this book

This book describes the IBM® WebSphere® MQ extended transactional clients, which are supplied for use with the WebSphere MQ Version 5.3 products. The book contains the following chapters:

Who this book is for

This book is for users of any of the following WebSphere MQ Version 5.3 products:

- WebSphere MQ for AIX®
- WebSphere MQ for HP-UX
- WebSphere MQ for iSeries™
- WebSphere MQ for Linux for Intel
- WebSphere MQ for Linux for zSeries™
- WebSphere MQ for Solaris
- WebSphere MQ for Windows®

What you need to know to understand this book

To understand this book, you need a good knowledge of the concepts and terminology associated with WebSphere MQ and practical experience in implementing WebSphere MQ, particularly in the following areas:

- WebSphere MQ clients
- Using an external transaction manager to coordinate updates to WebSphere MQ resources and those of other resource managers

You also need to be familiar with the operating systems and communications protocols you are using.

Terms used in this book

The term *Linux* denotes:

- Linux for Intel
- Linux for zSeries

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

- AIX
- HP-UX
- Linux for Intel
- Linux for zSeries
- Solaris

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

- Windows NT®

- Windows XP
- Windows 2000

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

The description of WebSphere MQ function in this book is in addition to, and must be used in conjunction with, the information provided in the latest editions of the other WebSphere MQ books. These books are provided in softcopy form (Adobe Acrobat PDF and HTML) on a separate CD in the product package.

Chapter 1. Introduction

This chapter reviews some concepts and terminology associated with transaction management, explains what a WebSphere MQ extended transactional client is, and specifies the supported operating environments for the extended transactional clients. It also summarizes the steps to implement an extended transactional client and introduces the remaining chapters in the book.

The chapter contains the following sections:

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.

What is an extended transactional 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.

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.

Supported operating environments

WebSphere MQ extended transactional clients are available for the platforms listed in Table 1. The table also lists the supported external transaction managers for each platform.

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

Extended transactional client platform	Supported external transaction managers
AIX	BEA Tuxedo, V6.4 and V6.5 TXSeries™, V4.3 and V5.0 WebSphere Application Server, V4.0 and V5.0
HP-UX	BEA Tuxedo, V6.4 and V6.5 TXSeries, V4.2 and V5.0 WebSphere Application Server, V4.0 and V5.0
Linux for Intel	WebSphere Application Server, V4.0 and V5.0
Linux for zSeries	WebSphere Application Server, V4.0 and V5.0
Solaris	BEA Tuxedo, V6.4 and V6.5 TXSeries, V4.3 and V5.0 WebSphere Application Server, V4.0 and V5.0
Windows systems	BEA Tuxedo, V6.4 and V6.5 Microsoft® Transaction Server (MTS)/COM+ TXSeries, V4.3 and V5.0 WebSphere Application Server, V4.0 and V5.0

Note that there is no extended transactional client for the following platforms:

- OS/400®
- Windows 98
- z/OS™

A client application that is using an extended transactional client can connect to a queue manager of the following WebSphere MQ Version 5.3 products only:

- WebSphere MQ for AIX
- WebSphere MQ for HP-UX
- WebSphere MQ for iSeries
- WebSphere MQ for Linux for Intel
- WebSphere MQ for Linux for zSeries
- WebSphere MQ for Solaris
- WebSphere MQ for Windows

Even though there is no extended transactional client that runs on OS/400, a client application that is using an extended transactional client can still connect to a queue manager that runs on OS/400.

For each platform, the hardware and software requirements for the extended transactional client are the same as those for the WebSphere MQ base client. For this information, therefore, see *WebSphere MQ Clients* or the Quick Beginnings book for your client platform. 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.

How do I implement an extended transactional client?

To implement an extended transactional client, you must do the following:

1. Install the extended transactional client.

   Chapter 2, "Installing an extended transactional client", on page 7 provides the installation instructions for each platform.

2. Configure the extended transactional client.

   To support client applications that are not written in Java, you configure the WebSphere MQ base client as described in *WebSphere MQ Clients*. The additional configuration tasks for the extended transactional function are described in Chapter 3, "Configuring an extended transactional client", on page 23.

   If you are using WebSphere MQ JMS and WebSphere Application Server, you configure WebSphere MQ Java for use in client mode as described in *WebSphere MQ Using Java*. The same book describes how to configure the interface between WebSphere MQ JMS and WebSphere Application Server to support 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. The additional configuration required for the extended transactional function is described in Chapter 3, "Configuring an extended transactional client", on page 23.

3. Prepare your client applications.

   For client applications that are not written in Java, you do this by compiling them and linking them with the appropriate client system libraries. You can then run the applications. This is described in Chapter 4, "Preparing and running client applications", on page 35.

   If you are using WebSphere MQ JMS and WebSphere Application Server, you prepare and run your client applications as described in *WebSphere MQ Using Java*. When you run the applications, you might see certain warning messages. These are documented in Chapter 4, "Preparing and running client applications", on page 35.

Chapter 2. Installing an extended transactional client

This chapter describes how to install the extended transactional client for each of the following platforms:

- AIX
- HP-UX
- Linux
- Solaris
- Windows systems

Before you install an extended transactional client, read the readme file for the latest information about the product.

On each 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.

The chapter contains the following sections:

Installing on AIX

To install the extended transactional client on AIX, you must first install the WebSphere MQ base client. You can install the WebSphere MQ base client using the WebSphere MQ Client 1 CD-ROM, or as part of a server installation using the WebSphere MQ for AIX Server CD-ROM. As a minimum, you must install the components of the WebSphere MQ base client that are listed in Table 2.

Table 2. Required components of the WebSphere MQ base client on AIX

Component	File set
Runtime	mqm.base.runtime
Client	mqm.client.rte
Java messaging	mqm.java.rte

For information about how to install the WebSphere MQ base client, see *WebSphere MQ Clients* or *WebSphere MQ for AIX, V5.3 Quick Beginnings*.

After you have installed the WebSphere MQ base client, you can install the extended transactional function. To install the extended transactional function, use the WebSphere MQ Extended Transactional Clients CD-ROM. The extended transactional function supports all applications, those written in Java and those not written in Java.

See the following sections for information about how to install, and uninstall, the extended transactional function:

- "Installing the extended transactional function"
- "Uninstalling the extended transactional function" on page 9

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.

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 *WebSphere MQ Clients* or *WebSphere MQ for AIX, V5.3 Quick Beginnings*.

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 extended transactional client on HP-UX, you must first install the WebSphere MQ base client. You can install the WebSphere MQ base client using the WebSphere MQ Client 1 CD-ROM, or as part of a server installation using the WebSphere MQ for HP-UX Server CD-ROM. As a minimum, you must install the components of the WebSphere MQ base client that are listed in Table 3.

Table 3. Required components of the WebSphere MQ base client on HP-UX

Component	File set
Runtime	MQSERIES.MQM-BASE
Client	MQSERIES.MQM-CL-HPUX
Java messaging	MQSERIES.MQM-JAVA

For information about how to install the WebSphere MQ base client, see *WebSphere MQ Clients* or *WebSphere MQ for HP-UX, V5.3 Quick Beginnings*.

After you have installed the WebSphere MQ base client, you can install the extended transactional function. To install the extended transactional function, use the WebSphere MQ Extended Transactional Clients CD-ROM. The extended transactional function supports all applications, those written in Java and those not written in Java.

See the following sections for information about how to install, and uninstall, the extended transactional function:

- "Installing the extended transactional function"
- "Uninstalling the extended transactional function" on page 11

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.

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 *WebSphere MQ Clients* or *WebSphere MQ for HP-UX, V5.3 Quick Beginnings*.

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 extended transactional client on Linux, you must first install the WebSphere MQ base client. You can install the WebSphere MQ base client using the WebSphere MQ Client 2 CD-ROM, or as part of a server installation using the WebSphere MQ for Linux for Intel Server CD-ROM or the WebSphere MQ for Linux for zSeries Server CD-ROM. As a minimum, you must install the components of the WebSphere MQ base client that are listed in Table 4.

Table 4. Required components of the WebSphere MQ base client on Linux

Component	Package name
Runtime	MQSeriesRuntime
Client	MQSeriesClient
Java messaging	MQSeriesJava

For information about how to install the WebSphere MQ base client, see *WebSphere MQ Clients* or *WebSphere MQ for Linux for Intel and Linux for zSeries, V5.3 Quick Beginnings*.

After you have installed the WebSphere MQ base client, you can install the extended transactional function. To install the extended transactional function, use the WebSphere MQ Extended Transactional Clients CD-ROM. The extended transactional function supports all applications, those written in Java and those not written in Java.

See the following sections for information about how to install, and uninstall, the extended transactional function:

- "Installing the extended transactional function"
- "Uninstalling the extended transactional function" on page 13

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
     ```

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 *WebSphere MQ Clients* or *WebSphere MQ for Linux for Intel and Linux for zSeries, V5.3 Quick Beginnings*.

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 extended transactional client on Solaris, you must first install the WebSphere MQ base client. You can install the WebSphere MQ base client using the WebSphere MQ Client 1 CD-ROM, or as part of a server installation using the WebSphere MQ for Solaris Server CD-ROM. As a minimum, you must install the `Sun Solaris client libraries` component of the WebSphere MQ base client. For information about how to install the WebSphere MQ base client, see *WebSphere MQ Clients* or *WebSphere MQ for Solaris, V5.3 Quick Beginnings*.

After you have installed the WebSphere MQ base client, you can install the extended transactional function. To install the extended transactional function, use the WebSphere MQ Extended Transactional Clients CD-ROM. The extended transactional function supports all applications, those written in Java and those not written in Java.

See the following sections for information about how to install, and uninstall, the extended transactional function:

- "Installing the extended transactional function"
- "Uninstalling the extended transactional function" on page 15

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 15 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.

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 *WebSphere MQ Clients* or *WebSphere MQ for Solaris, V5.3 Quick Beginnings*.

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 systems

To install the extended transactional client on Windows systems, you must first install the WebSphere MQ base client. You can install the WebSphere MQ base client as part of a client installation using the WebSphere MQ Client 1 CD-ROM, or as part of a server installation using the WebSphere MQ for Windows Server CD-ROM. As a minimum, you must install the Windows Client feature of the WebSphere MQ base client but, if you want to use WebSphere MQ JMS, you must install the Java Messaging feature as well. For information about how to install the WebSphere MQ base client, see *WebSphere MQ Clients* or *WebSphere MQ for Windows, V5.3 Quick Beginnings*.

After you have installed the WebSphere MQ base client, you can install the extended transactional function. To install the extended transactional function, use the WebSphere MQ Extended Transactional Clients CD-ROM. The extended transactional function supports all applications, those written in Java and those not written in Java.

See the following sections for information about how to install, and uninstall, the extended transactional function:

- "Installing the extended transactional function"
- "Other ways of installing the extended transactional function" on page 17
- "Uninstalling the extended transactional function" on page 19

To install the extended transactional function, you must be logged on to Windows as an administrator.

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 ways of installing the extended transactional function

This section describes the following alternative ways of installing the extended transactional function on Windows systems:

Installing 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 16 from step 2 on page 16 to the end.

Unattended, or silent, installation

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 "X:\Windows\MSI\IBM WebSphere MQ Extended Transactional Client.msi" /q
TRANSFORMS=:1033 AGREETOLICENSE="yes"
```

where *X* is the drive letter of your CD-ROM drive and /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, V5.3 Quick Beginnings*.

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

You cannot use a response file.

Installing using Microsoft System Management Server

There are two major steps to install the extended transactional function using Microsoft System Management Server (SMS):

1. Create an SMS software package (see "Creating an SMS software package").
2. Create and run an SMS job to distribute and install the package (see "Creating and running an SMS job" on page 19).

For more detailed information about how to create a software package, and to create and run a job, refer to the SMS documentation.

Creating an SMS software package: To create an SMS software package, do the following:

1. From the Microsoft SMS Administrator application, open the **Packages** folder and create a new package.
2. In the SMS **Package Properties** dialog, click **Import** to create the software package by importing a Package Definition File (PDF).
3. In the **File Browser** dialog, select the drive where the WebSphere MQ Extended Transactional Clients CD-ROM is located.
4. Select the Windows folder, which contains the package definition file WebSphere MQ.pdf.

   You can also find the WebSphere MQ.pdf file in the local drive, or the shared network drive to where you copied the installation software for the extended transactional client.
5. Select the **WebSphere MQ.pdf** file and click **OK**.

6. Click **Workstation**. In the **Source Directory** entry field, specify the fully qualified path name to the WebSphere MQ root folder that contains the WebSphere MQ installation software.
7. Select the appropriate Workstation Command Line:
   - **Automated Uninstallation of IBM WebSphere MQ Extended Transactional Client**
   - **Automated Installation of IBM WebSphere MQ Extended Transactional Client (US English)**
8. Click **Properties** for each process and review the **Command Line** entry field to ensure that the parameters are what you require.
9. Click **Close** to close the **Workstation Properties** dialog.

   Note: If you specified a local path in the **Source Directory** entry field, you will get a pop-up dialog warning you that the local path you specified might not be accessible to SMS components running on another machine. Click **OK** to continue.
10. Click **OK** to close the **Package Properties** window.

    A pop-up dialog appears indicating that SMS will update the software package at all sites. Click **OK** to continue.

The software package has been created and can be installed by creating an SMS job.

Creating and running an SMS job: You must now create and run an SMS job to distribute and install the software package that contains the WebSphere MQ installation software.

When creating an SMS job, ensure that you select the appropriate workstation command. The workstation commands are displayed on the **Job Details** dialog in the **Run Phase** section and appear in a drop-down list.

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 *WebSphere MQ Clients* or *WebSphere MQ for Windows, V5.3 Quick Beginnings*.

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" on page 19 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 3. 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

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 *WebSphere MQ Clients*. 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 is restriction 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 25.

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 28.

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 29
- "Configuring for Encina" on page 30
- "Configuring for Tuxedo" on page 31

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 27.

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 5 for the assumed values of the AXLIB parameter.

Table 5. 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

Configuring an extended transactional client

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 6.

Table 6. 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 7 for the fully qualified path name of each switch load file.

Table 7. 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:

Configuring an extended transactional client

```
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 8. 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 8. 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 9 for this information.

Table 9. 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.

Configuring an extended transactional client

Chapter 4. Preparing and running client applications

This chapter contains the following sections:

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 10, for AIX, HP-UX, and Solaris, and Table 11, 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 10. 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 11. 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 12 on page 36 lists the CICS, Encina, and Tuxedo sample programs that are supplied for use on AIX, HP-UX, and Solaris client systems. Table 13 on page 36 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*.

Preparing and running client applications

Table 12. 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 13. 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

Appendix. 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.

Notices

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.

COPYRIGHT LICENSE:

This information contains sample application programs in source language, which illustrate programming techniques on various operating platforms. You may copy, modify, and distribute these sample programs in any form without payment to IBM, for the purposes of developing, using, marketing or distributing application programs conforming to the application programming interface for the operating platform for which the sample programs are written. These examples have not been thoroughly tested under all conditions. IBM, therefore, cannot guarantee or imply reliability, serviceability, or function of these programs. You may copy, modify, and distribute these sample programs in any form without payment to IBM for the purposes of developing, using, marketing, or distributing application programs conforming to IBM's application programming interfaces.

Trademarks

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

AIX	CICS	DB2
Encina	IBM	iSeries
OS/400	TXSeries	WebSphere
z/OS	zSeries	

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

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

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

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

Index

T

U

W

X

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.

Printed in U.S.A.

SC34-6275-00