WebSphere MQ Everyplace

Introduction

Version 2.0

SC34-6277-00

Take Note!

Before using this information and the product it supports, read the general information under “Notices” on page 39

First Edition (November 2002)

This edition applies to WebSphere® MQ Everyplace™ Version 2.0 (Program number: 5724-C77) and to all subsequent releases and modifications until otherwise indicated in new editions.

This document is continually being updated with new and improved information. For the latest edition, please see the WebSphere MQ family library Web page at *http://www.ibm.com/software/mqseries/library/*.

Contents

About this book

This book is a general introduction to the WebSphere MQ Everyplace product. It covers the product concepts and its relationship to other WebSphere MQ products.

For WebSphere MQ Everyplace installation procedures see *WebSphere MQ Everyplace Read Me First*.

For detailed information on the WebSphere MQ Everyplace Java™ API and the C codebase, and how to use it to create WebSphere MQ Everyplace applications, see the *WebSphere MQ Everyplace Application Programming Guide*, the *WebSphere MQ Everyplace Java Programming Reference*, and the *WebSphere MQ Everyplace C Programming Reference* on the product CD.

For detailed information on the C Bindings and their use, see *WebSphere MQ Everyplace C Bindings Programming Reference* and *WebSphere MQ Everyplace C Bindings Programming Guide*.

For information relating to using WebSphere MQ Everyplace on the Palm platform see *WebSphere MQ Everyplace C Programming Guide for Palm OS*.

This document is continually being updated with new and improved information. For the latest edition, please see the WebSphere MQ family library Web page at *http://www.ibm.com/software/mqseries/library/*.

License warning

WebSphere MQ Everyplace is a toolkit that enables users to write WebSphere MQ Everyplace applications and to create an environment in which to run them. Before deploying this product, or applications that use it, please make sure that you have the necessary licenses.

1. The pricing of licenses for use of the Program on servers is based on 'Processor License Units'. Use of each copy of the Program on a server requires one Processor License Unit to be acquired for each processor or symmetric multiprocessor contained in the server on which the copy of the Program is to run. Different types of Processor License Units and 'Device Use Authorisations' are required, depending on whether the Program is running on point-of-sale, that is retail, equipment or on another type of computer. Use of the Program on retail equipment requires a 'Retail' server license, whereas use on other (non-retail) equipment requires a 'Network' server license.
2. Additional 'Device Use Authorisation' is required for any use of the Program on a separate client device, except those included in the Network Server license, as described at 3) below.
3. Each 'Network' server license includes authorisation for the restricted use of the Program with no more than one hundred (100) client devices, on condition that all such copies are used in the same economic enterprise or organisation as the server copy.

Please refer to *http://www.ibm.com/software/mqseries* for details of these restrictions.

Who should read this book

This book is for those interested in using secure messaging on lightweight devices such as sensors, phones, Personal Digital Assistants (PDAs), and laptop computers, and those with a need to extend the scope of a WebSphere MQ Everyplace messaging network.

Prerequisite knowledge

An understanding of the concepts of secure messaging is an advantage.

If you do not have this understanding, you might find it useful to read the following WebSphere MQ book:

- *An Introduction to Messaging and Queuing*, GC33-0805

This book is available in softcopy form from the Book section of the online WebSphere MQ library. This can be reached from the WebSphere MQ Web page: *http://www.ibm.com/software/mqseries/library/*.

Terms used in this book

The following terms are used throughout this book:

WebSphere MQ family
Refers to the following WebSphere MQ products:

- **WebSphere MQ Workflow** simplifies integration across the whole enterprise by automating business processes involving people and applications.
- **WebSphere MQ Integrator** is a powerful message-brokering software that provides real-time, intelligent, rules-based message routing, and content transformation and formatting.
- **WebSphere MQ Messaging** provides any-to-any connectivity from desktop to mainframe, through business quality messaging, with over 35 platforms supported.

WebSphere MQ Messaging
Refers to the following messaging product groups:

- **Distributed messaging:** WebSphere MQ for Windows NT and Windows 2000, AIX®, iSeries®, HP-UX, Solaris, and other platforms
- **Host messaging:** WebSphere MQ for z/OS®
- **Pervasive messaging:** WebSphere MQ Everyplace

WebSphere MQ
Refers to the following Messaging product groups:

- Distributed messaging
- Host messaging

WebSphere MQ Everyplace
Refers to the WebSphere MQ pervasive messaging product group.

Client A run-time component that provides access to queuing services on a server for local user applications.

Device platform
A small computer that is capable of running WebSphere MQ Everyplace only as a client.

Server A device that has a WebSphere MQ Everyplace connection manager and that responds to to requests for information in a client/server set-up.

Server platform
A computer of any size that is capable of running WebSphere MQ Everyplace as a server or client.

Gateway
A computer of any size running WebSphere MQ Everyplace programs that include the WebSphere MQ bridge function.

Summary of Changes

This section describes changes to this edition of the *WebSphere MQ Everyplace Introduction*. Changes since the previous edition of the book are marked by vertical lines to the left of the changes.

Changes for this edition (SC34-6277-00)

In addition to editorial corrections and improvements to the text, the following information has been modified or added:

- Product version number.
- Migration information.
- The version 1.2.7 Programming Guide has been replaced with:
  - The WebSphere MQ Everyplace Application Programming Guide.
  - The WebSphere MQ Everyplace System Programming Guide.
  - The WebSphere MQ Everyplace Configuration Guide.
- The version 1.2.7 Native Client Information manual is now called the WebSphere MQ Everyplace C Programming Guide for Palm OS.

Chapter 1. Overview

WebSphere MQ Everyplace is a member of the IBM WebSphere MQ family of business messaging products. It exchanges *messages* with various applications, providing once and once-only assured delivery. WebSphere MQ Everyplace is designed to integrate well with other members of the WebSphere MQ family, and other components of the WebSphere MQ Everyplace Server.

WebSphere MQ Everyplace is designed to satisfy the messaging needs of lightweight devices, such as sensors, phones, Personal Digital assistants (PDAs), and laptop computers. It supports mobile environments and is suitable for use over public networks, supporting requirements that arise from the use of fragile communication networks. As many WebSphere MQ Everyplace applications run outside the protection of an Internet firewall, it also provides security capabilities.

Chapter 2. Product concepts

This chapter introduces WebSphere MQ Everyplace under the following headings:

- The WebSphere MQ Family
- How WebSphere MQ Everyplace extends the WebSphere MQ family
- Customer requirements

The WebSphere MQ Family

The WebSphere MQ family includes many products, offering a range of capabilities, as illustrated in Figure 1.

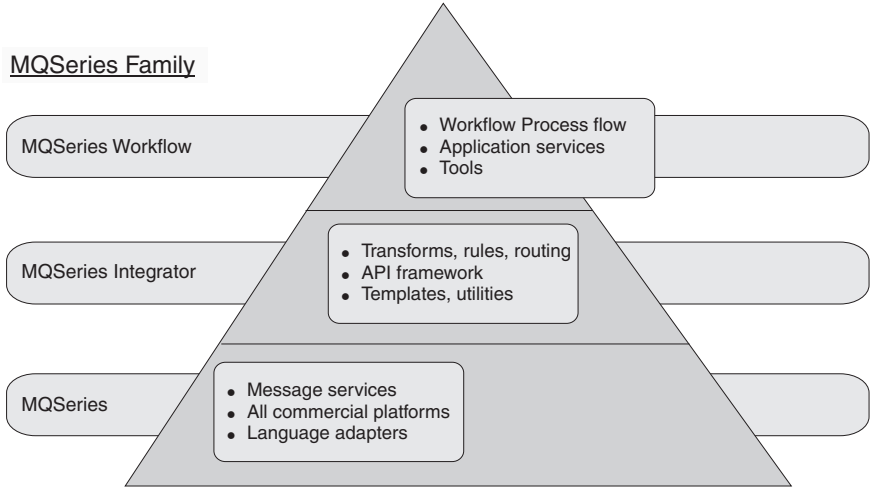


Figure 1. The WebSphere MQ family

- **WebSphere MQ Workflow** simplifies integration across the whole enterprise by automating business processes involving people and applications.
- **WebSphere MQ Integrator** is powerful message-brokering software that provides real-time, intelligent rules-based message routing, and content transformation and formatting.
- **WebSphere MQ Messaging** provides any-to-any connectivity from desktop to mainframe, through business quality messaging, supporting over 35 platforms.

Both WebSphere MQ Workflow and WebSphere MQ Integrator products take advantage of the connectivity provided by the WebSphere MQ messaging layer.

Basic Messaging

Messaging, irrespective of the particular product or product group, is based on *queues* and *queue managers*. Queue managers manage queues that can store messages. Applications communicate with a *local queue manager*, and *get* or *put* messages to

queues. If a message is put to a remote queue (a queue owned by another queue manager), the message is transmitted over *connections* to the *remote queue manager*. In this way, messages can hop through one or more intermediate queue managers before reaching their destination. The essence of messaging is to uncouple the sending application from the receiving application, queuing messages at intermediate points, if necessary.

WebSphere MQ and WebSphere MQ Everyplace supply WebSphere MQ family messaging. Both are designed to support one or more hardware server platforms and most associated operating systems. Given the wide variety in platform capabilities, these individual products are organized into product groups, reflecting common function and design:

- **Distributed messaging:** WebSphere MQ for Windows NT®, Windows® 2000, AIX, iSeries™, HP-UX, Solaris, and other platforms
- **Host messaging:** WebSphere MQ for z/OS™
- **Pervasive messaging:** WebSphere MQ Everyplace for Windows, AIX, Solaris, Linux, and HP-UX

WebSphere MQ host and distributed products

WebSphere MQ host and distributed messaging products are used to support many different network configurations, all of which involve *clients* and *servers*, some examples of which are illustrated in Figure 2.

Note: The terms client and server have very specific meanings within WebSphere MQ host, distributed, and workstation messaging products, that do not always correspond to their meaning within WebSphere MQ Everyplace.

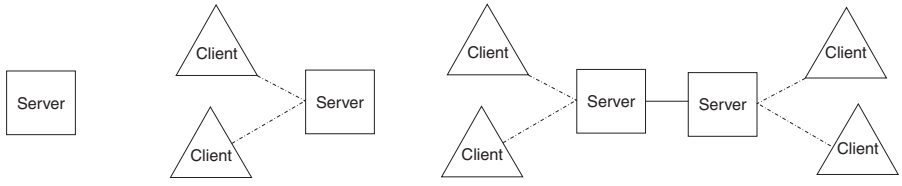


Figure 2. Simple host and distributed configurations

In Figure 2:

a) Standalone server

A queue manager runs on a single server. One or more applications run on that server, exchanging messages using queues.

b) Client/server

A queue manager runs on a server, but the clients each have access to it through a bidirectional connection called a *client channel*. The client channel implements something similar to a remote procedure call (RPC). Applications can run on the clients, accessing server queues. One advantage of the

client/server configuration is that the client-messaging infrastructure is lightweight, because it depends on the server queue manager. One disadvantage is that clients and their associated server operate synchronously and, therefore, require the client channel to be available at all times.

c) Distributed client/server

A distributed client/server configuration involves multiple servers. In this case, servers exchange messages through unidirectional connections called *message channels*. Message channels assure safe and asynchronous exchange of message data. Message channels do not need to be available for the clients to continue processing. However, no messages can flow between servers when there are no communication links established between servers.

WebSphere MQ Everyplace

WebSphere MQ Everyplace supports a variety of network configurations. There is no concept of a client or a server as in the WebSphere MQ host or distributed products. Instead, you can configure WebSphere MQ Everyplace *queue managers* to act as clients or servers, enabling them to perform application-defined tasks.

An example of tailored configuration is that you can give WebSphere MQ Everyplace the ability to exchange messages with WebSphere MQ host queue managers. To do this, configure a WebSphere MQ Everyplace queue manager with *bridge* capabilities. Without the bridge, a queue manager can communicate directly with other WebSphere MQ Everyplace queue managers only. However, it can communicate indirectly through other queue managers in the network that have bridge capabilities.

Note: A new node for WebSphere MQ Integrator (MQSI) allows you to connect to WebSphere MQ Everyplace, without using the WebSphere MQ bridge.

How WebSphere MQ Everyplace extends the WebSphere MQ family

This section describes some requirements that have shaped the design and implementation of WebSphere MQ Everyplace.

Capabilities

WebSphere MQ Everyplace extends the messaging scope of the WebSphere MQ family by:

- Supporting low-end devices, such as PDAs, telephones, and sensors. WebSphere MQ Everyplace also supports intermediate devices such as laptops, workstations, distributed, and host platforms. WebSphere MQ Everyplace offers once and once-only assured delivery of messages, and permits message exchange with other family members.
- Offering lightweight messaging facilities.
- Providing extensive security features to protect messages, queues, and related data, whether in storage or in transmission.
- Operating efficiently in hostile communications environments where networks are unstable, or where bandwidth is tightly constrained. WebSphere MQ Everyplace has an efficient wire protocol and automated recovery from communication link failures.

- Supporting the mobile user, allowing network connectivity points to change as devices roam. WebSphere MQ Everyplace also allows control of behavior in conditions where battery resources and networks are constrained.
- Operating through suitably configured firewalls.
- Minimizing administration tasks for the user. This makes WebSphere MQ Everyplace a suitable base on which to build utility-style applications.
- Being easily customized and extended, through the use of application-supplied *rules*

Customer requirements

WebSphere MQ Everyplace supports mobility and fragile communication networks. Because WebSphere MQ Everyplace is targeted at lightweight devices, it is frugal in its use of system resources. It offers tailored functions and interfaces and does not aim to provide exactly the same capabilities as other members of the WebSphere MQ family. It also includes unique functions to support its particular classes of user, such as comprehensive security provision, messages, synchronous and asynchronous messaging, remote queue access, and message push and pull.

Applications

There are various possible WebSphere MQ Everyplace applications, many of which are expected to be custom applications developed for particular user groups. The following list gives some examples:

Retail applications

Trickle feeding till transactions to host systems, such as message brokers

Consumer applications

- Supermarket shopping from home using a PDA
- Gathering traveller preferences on airlines
- Financial transactions from a mobile phone

Control applications

- Collection and integration of data from oil pipeline sensors transmitted via satellite
- Remote operation of equipment (such as valves) with security to guarantee the validity of the operator

Mobile workforce

- Visiting professionals, for example an insurance agent
- Rapid publication of proof of customer receipt for parcel delivery companies
- Information exchange between kitchen and waiting staff
- Golf tournament score keeping
- Secure mobile systems messaging for the police
- Job information for utility workers in situations where communication is frequently lost
- Domestic meter reading

Personal productivity

- Mail and calendar replication
- Database replication
- Downloading to laptops

Chapter 3. Elements of WebSphere MQ Everyplace

The fundamental elements of the WebSphere MQ Everyplace programming model are messages, queues, and queue managers. WebSphere MQ Everyplace messages contain application-defined content. Messages are stored in a queue and can be moved across a WebSphere MQ Everyplace network. You can address messages to a target queue by specifying the target queue manager and queue name. Applications place messages on queues through a *put* operation and typically retrieve them through a *get* operation. Queues can either be *local* or *remote* and are managed by queue managers. The *registry* stores configuration data.

This chapter describes the following concepts:

- Messages
- Queues
- Queue Managers
- Administration
- Connections
- Adapters
- Dialup connection management
- Trace
- Event log
- Security
- Customizing rules
- Classes

Messages

A message is simply a collection of data sent by one application and intended for another application. WebSphere MQ Everyplace messages differ from those supported by WebSphere MQ messaging. In WebSphere MQ, messages are byte arrays, divided into a message header and a message body. WebSphere MQ creates the message header, which contains vital information, such as the identity of the reply-to queue, the reply-to queue manager, the message ID, and the correlation ID. The message body contains data that is useful only to the application.

Messages in WebSphere MQ Everyplace have no concept of a header or a message body. They are *MQeFields*, which consist of a name, a *data type*, and the data itself. Message names are ASCII character strings of unlimited length, excluding any of the characters:
{ } [ ] # ( ) : ; , ' " =
Table 1 describes the different data types:

Table 1. Data types

Type	Description
ASCII	String or a dynamic array of invariant ASCII strings, excluding any of the characters { } [ ] # ( ) : ; , ' " and =
Boolean	True or false value
Byte	Fixed array, or a dynamic array of byte values
Double floating point	Value, fixed array, or a dynamic array of double floating point values
Fields	Object or a dynamic array of fields objects (thus nesting of fields objects is supported)
Floating point	Value, fixed array, or a dynamic array of floating point values
Integer	4 byte value, fixed array, or a dynamic array of integers
Long integer	8 byte value, fixed array, or a dynamic array of long integers
Short integer	2 byte value, fixed array, or a dynamic array of short integers
Unicode	String or a dynamic array of Unicode strings

Additionally, messages include a UID (unique identifier) which is generated by WebSphere MQ Everyplace. This UID uniquely identifies each individual message object in the entire WebSphere MQ Everyplace network and is constructed from the:

Originating queue manager
: This is the name of the originating queue manager, which must be unique. It is added by the queue manager on receipt of the message. As it is ASCII, every character is one byte long.

Creation time
: This is the *timestamp* of a message. Therefore, in Java, this is the time that the message was created, and in C, this is the time that a fields item is added to a queue. The field item then becomes a message.

A message destined for another WebSphere MQ Everyplace queue manager does not require any additional information, though other properties are almost certainly present. Additional properties can:

- Reflect current status
- Be associated with a particular message subclass
- Allow you to customize a message

Note: In the C codebase a fields item only becomes a message upon arrival on a queue.

WebSphere MQ Everyplace adds property related information to a message (and subsequently removes it) in order to implement messaging and queuing operations. When sending a message between queue managers, you can add resend information to indicate that data is being retransmitted. Chapter 4, *Messaging*, of the WebSphere MQ Everyplace Application Programming Guide provides more information on message object properties.

Messages can also have *attributes*. Attributes are fundamental to the WebSphere MQ Everyplace security model and allow selective access to content and the protection of content. They have the following properties:

Table 2. Attribute object properties

Property	Description
Authentication	Controls access
Encryption	Protects the contents when the object is dumped (and allows restoration)
Compression	Reduces storage requirements (for transmission and storage)
Rule (Not applicable to C codebase)	Controls permitted operations

For more information on the properties in Table 2, see “Security” on page 21.

Queues

Queues are typically used to hold messages pending their removal by application programs. Each queue belongs to a queue manager. Applications are not normally permitted to directly access a queue. Instead, the queue manager acts as an intermediary between application programs and queues.

Queues are identified by name, and the name can be an ASCII character string of unlimited length, excluding characters:
{ } [ ] # ( ) : ; , ’ ″ =
However, queue names must be unique within a particular queue manager. For interoperability with WebSphere MQ, we recommend that you also observe WebSphere MQ naming restrictions, including a maximum name length of 48 characters. The name length may also be restricted by the file system you are using. WebSphere MQ Everyplace supports a number of different queue types:

Local queues

Applications use local queues to store messages in a safe and secure manner (excluding hardware failure or loss of the device). Local queues belong to a specific queue manger. This can be either a standalone queue manager or a queue manager that is connected to a network.

Remote queues

Remote queues are local references to queues that reside on another queue manager in the WebSphere MQ Everyplace network. The local reference has the same name as the target queue, but the remote queue definition identifies

the owning queue manager of the real queue. Remote queues also have properties concerned with access, security characteristics, and transmission options. Their mode of access can be either synchronous or asynchronous.

Store-and-forward queues

A store-and-forward queue stores messages on behalf of one or more queue managers until they are ready to receive them. This type of queue is not present in the C codebase. Store-and-forward queues have two main uses:

1. To enable the intermediate storage of messages in a network, so that they can proceed to their destination (a forwarding role)
2. To hold messages awaiting collection by a Home-server queue.

This type of queue is normally (but not necessarily) defined on a server or gateway. Store-and-forward queues can hold messages for many target queue managers, or there may be one store-and-forward queue for each target queue manager.

Home-server queues

While remote queues and store-and-forward queues push messages across the network, with the sending queues initiating the transmission, home-server queues pull messages from a remote queue. Messages are never addressed to a home-server queue.

A home-server queue definition identifies a store-and-forward queue on a remote queue manager. The home-server queue then pulls messages that are destined for its local queue manager from the store-and-forward queue. Multiple home-server queue definitions can be defined on a single queue manager, where each one is associated with a different remote store-and-forward queue.

Administration queues

An administration queue is a type of local queue that accepts administration messages. An administration message contain instructions, processed internally by the application, relating to a particular element of WebSphere MQ Everyplace. Each administration action can, optionally, cause an administration reply message to be sent back to the originating application. These reply messages inform you of the success or failure of the administration action. In this way, using administration queues allows an element on one queue manager to control the configuration of a second queue manager, either synchronously or asynchronously. Administration messages are processed in order of arrival on the administration queue. For further information, refer to the section on “Administration” on page 15.

WebSphere MQ bridge queues

A bridge queue is a specialist form of remote queue, describing a queue on a WebSphere MQ remote queue manager. Bridge queues put or get from the WebSphere MQ queue they reference. In Java only, it uses a transformer to perform any necessary data or message reformatting as each message is exchanged between the WebSphere MQ Everyplace and WebSphere MQ systems. You can only create a bridge queue on a gateway queue manager.

The WebSphere MQ Everyplace Application Programming Guide and the WebSphere MQ Everyplace Configuration Guide provide more information on queues.

WebSphere MQ Everyplace stores data securely on queues, ensuring that messages are physically written to the media and not simply stored by the operating system. However, WebSphere MQ Everyplace does not independently log changes to messages and queues. Therefore, to recover from media failure, you need to deploy hardware solutions, such as RAID disk systems. Alternatively, map the queue into recoverable storage, for example database subsystems.

WebSphere MQ Everyplace has four commonly used system queues:

Administration queue
Receives administration messages

Dead letter queue
Stores messages that cannot otherwise be delivered

Administration reply queue
Receives replies to administration messages (optional)

SYSTEM.DEFAULT.LOCAL.QUEUE
Shares a common name with the mandatory system queue on WebSphere MQ servers

Queue managers

The WebSphere MQ Everyplace queue manager allows WebSphere MQ Everyplace to support a variety of network configurations. It provides:

- A central point of access to a messaging and queueing network for WebSphere MQ Everyplace applications
- Optional client-side queuing
- Connection control
- Optional administration functions
- Once and once-only assured delivery of messages
- Automated recovery from failure conditions
- Customizable rules-based behavior

In WebSphere MQ Everyplace, you can only have one queue manager active on a single Java virtual machine (JVM), or in a single native application process at any one time. To have multiple queue managers on a machine, you require either multiple JVM or multiple native application processes.

Queue managers are identified by a globally unique name and an ASCII character string of unlimited length, excluding any of the following characters:
{ } [ ] # ( ) : ; , ' " =
This restriction is not enforced by WebSphere MQ Everyplace or WebSphere MQ, but duplicate queue manager names may cause messages to be delivered to the wrong

queue manager. For interoperability, we recommend that you limit the maximum name length to 48 characters. The file system that you are using may also restrict the name length.

You can configure queue managers with or without local queueing. All queue managers support synchronous messaging operations. A queue manager with local queueing also supports asynchronous message delivery. Asynchronous message delivery and synchronous message delivery have very different characteristics and consequences:

Synchronous message delivery
With synchronous message delivery the application puts the message to WebSphere MQ Everyplace for delivery to the remote queue. WebSphere MQ Everyplace simultaneously contacts the target queue and delivers the message. After delivery, WebSphere MQ Everyplace returns immediately to the application. If the message cannot be delivered, the sending application receives immediate notification. WebSphere MQ Everyplace does not assume responsibility for message delivery in the synchronous case (non-assured message delivery).

Asynchronous message delivery
With asynchronous message delivery the application puts the message to WebSphere MQ Everyplace for delivery to a remote queue. WebSphere MQ Everyplace immediately returns to the application. If the message can be delivered immediately, or moved to a suitable staging post, then it is sent. If not, it is stored locally. Asynchronous delivery provides once, and once-only assured delivery, because the message has been passed to WebSphere MQ Everyplace and it has become responsible for delivery (assured message delivery).

Chapter 6, *Message delivery*, of the WebSphere MQ Everyplace Application Programming Guide provides detailed information on synchronous and asynchronous messaging.

Queue manager configuration

The queue manager runs in an environment established by WebSphere MQ Everyplace, before the queue manager is loaded. The queue manager stores its configuration information in its registry. “The registry” on page 22 provides more information on this. The queues themselves (containing messages) are stored in queue stores.

You can configure the WebSphere MQ Everyplace environment using the API, utilities shipped with WebSphere MQ Everyplace, or management tools such as MQe_Explorer. These methods can capture the environment parameters in an initialization file, but this is optional. Chapter 5, *Queue manager operations*, of the WebSphere MQ Everyplace Application Programming Guide provides information on how to configure queue managers. The WebSphere MQ Everyplace Configuration Guide provides information on configuring queue managers using the MQe_Explorer tool.

You can configure a queue manager with WebSphere MQ bridge capabilities. This is called a gateway and, in Java, it exchanges messages with WebSphere MQ host and

distributed products. The C codebase uses a device queue manager only. Chapter 7, *Interoperability with other messaging systems* of the WebSphere MQ Everyplace Application Programming Guide provides detailed information on how to configure the bridge.

Queue manager operations

Queue managers support messaging operations and manage queues. Applications access messages through the services of the queue manager using methods such as:

Get This operation removes messages from a queue.

Put This operation places messages on a queue.

Delete By specifying the UID, you can delete messages from a queue without using the get operation.

Browse
You can browse queues for messages using a *filter*. Browsing retrieves all the messages that match the filter, but leaves them on the queue. WebSphere MQ Everyplace also supports *Browsing under lock.* This allows you to lock the matching messages.

Wait In Java, applications can *wait* for a specified time for messages to arrive on a queue. This does not apply to the C codebase.

Listen In Java, applications can listen for WebSphere MQ Everyplace message events, again with an optional filter. However, in order to do this, you must add a listener to the queue. Listeners are notified when messages arrive on a queue. This does not apply to the C codebase.

Many of these operations take a *filter* as one of their parameters. A filter matches an element for equality and any parts of the message can be used for selective retrieval. Most method calls also include an attribute to be used in the encoding or decoding of a message. Chapter 4, *Messaging*, of the WebSphere MQ Everyplace Application Programming Guide provides detailed information on messaging operations.

Administration

The WebSphere MQ Everyplace interface handles the generation and receipt of administration messages, enabling administration. While applications are responsible for message-related functions, administration provides facilities to configure and manage WebSphere MQ Everyplace resources such as queues and connections.

Requests are sent to the administration queue of the target queue manager and replies can be received, if required. Any local or remote WebSphere MQ Everyplace application program can create and process administration messages directly or indirectly through helper methods.

You can perform some administration actions using the administrator. These actions are performed only on resources that are managed by the local queue manager.

The administration queue itself cannot perform the administration of individual resources. The relevant information is contained in each resource and its corresponding administration message.

Administration messages

Once created, queue managers are configured through the sending of administration messages to the target queue manager administration queue. A queue manager that does not have an administration queue cannot be administered. The intent behind using administration messages is that both local and remote administration is performed in an identical manner.

An administration message is created and sent to the administration queue of the queue manager to be administered. You can apply queue-based security attributes to control access. An administration message includes details of the request, indicates whether or not a response is required, and contains the address identifying the target queue manager and queue. Therefore, WebSphere MQ Everyplace has the following styles of administration message:

- Commands that indicate an administration action that does not require a reply
- Requests that require a reply
- Reply messages constructed from a copy of the original message

The sender can add additional fields for use by the receiver. The administration queue itself acts upon the message. Administration messages can inquire on, create, delete, or update objects. For a subset of the objects they can perform additional functions, such as stop and start. The WebSphere MQ Everyplace Configuration Guide contains more detailed information on administration messages.

Administration messages can also be generated indirectly through the MQe_Explorer, a management tool that provides a graphical user interface for system administration. MQe_Explorer is not included with WebSphere MQ Everyplace but is available for free download from the WebSphere MQ Everyplace Web site:
http://www.ibm.com/software/ts/mqseries/everyplace

Selective administration

The authenticator on the administration queue can control access to administration. The supplied authenticator considers local applications to represent the same local user and, therefore, either enables or prevents administration for all of the applications.

Starting the authenticator on the connection, before any administration messages flow, controls remote administration applications. This distinguishes different remote applications from each other, and then enables or prevents administration for each remote application. In all cases, administration is either completely enabled or prevented.

An authenticator can keep track of permissions associated with user identities, and administration messages can subsequently be processed on the basis of these permissions. “Security” on page 21, provides more information on authentication. You

can also use rules that are associated with queues to enable or prevent actions in a similar manner. “Customizing Rules” on page 24, provides more information on rules.

Monitoring and related actions

Administration involves more than creating and modifying elements. It can include monitoring a system and informing an operator when a queue is full, or dealing with an error situation, for example taking appropriate action when a message arrives that is too large for its target queue. WebSphere MQ Everyplace handles these aspects using rules, whenever elements significantly change their status or when certain types of error situations arise. WebSphere MQ Everyplace provides a default rule implementation, which users can customize if they wish. “Customizing Rules” on page 24, provides more information on this.

Connections

A connection provides a queue manager with information to establish communication links with a remote queue manager. Queue managers then use connections to exchange information. Connection definitions are stored locally at each queue manager.

Note: The C codebase is a device queue manager only.

Some of the key features of connections are:

Support for both *synchronous* and *asynchronous* messaging
Synchronous messaging provides a transmission service directly from the source application to the target queue, without queuing at the source queue manager. Asynchronous messaging is a transmission service from the source queue manager to the target queue, with possible queuing at the source queue manager.

***End-to-end service* provision**
Connections go from the source queue manager to a destination queue manager, possibly running through intermediate queue managers. The underlying transport protocol used can change as the connection passes through these intermediates. Several connections can link together to form end-to-end connections.

Support for *compression*, *encryption*, and *authentication*
Connections have these security characteristics to protect the data in transit.

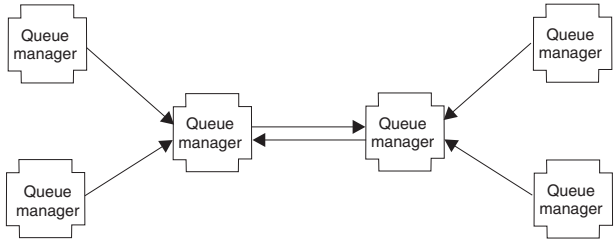
Support for *client/server* operation
Client/server connections are request/response. The client makes a request of the server and the server responds to that request. Note that this does not restrict the message flow. Messages can flow from client to server and from server to client. Chapter 4, *Message delivery*, of the WebSphere MQ Everyplace Application Programming Guide provides detailed information on client/server connections.

The following diagrams show some typical WebSphere MQ Everyplace configurations. For the purpose of clarity, the diagrams show only the direct connections that have been defined. You can also define indirect connections that exploit the direct

connections. In the diagrams, a line with the arrow pointing to the server represents a client/server connection. Clients can use the client/server connection both to send messages to the server and to pull messages destined for themselves from that server. Lines with no arrows indicate WebSphere MQ client channels that enable communications between WebSphere MQ Everyplace and WebSphere MQ.

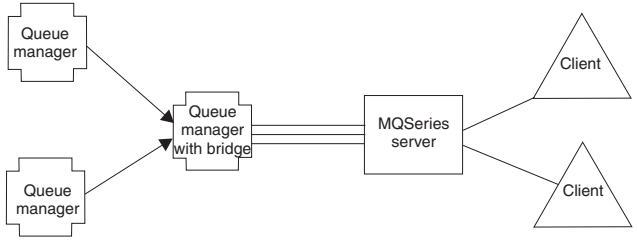
(a) Stand-alone queue manager

(a) Shows a standalone queue manager being used to support one or more applications that use queues to exchange data.



(d) A small network

(b) Shows a small network configuration, where the central server queue managers use a pair of direct client/server connections to exchange information. Each client queue manager uses a direct client/server connection to link to one of the server queue managers. A single queue manager can initiate either client or server connections, and respond as a server. In this case, the listener and the standard listener must have different port numbers.



(e) An integrated MQSeries family network

(c) A WebSphere MQ Everyplace configuration where one of the queue managers has been configured with the bridge option and the pool of client channels has been directed at a single target WebSphere MQ host/distributed server.

As connections typically define the access to a remote queue manager, they are sometimes referred to as remote queue manager definitions.

You can also specify indirect connections. In this case, WebSphere MQ Everyplace routes the connection through other queue managers (which can be chained), and the

protocol can change en route. Indirect connections are particularly useful in enabling devices to have a single point of entry to a WebSphere MQ Everyplace network.

As with most WebSphere MQ elements, you can define aliases for connections. Use a local connection, defined as a connection with a name matching that of the local queue manager, to define alias names for the local queue manager itself.

Connections support bidirectional flows and are established by the queue manager as required. Asynchronous and synchronous messaging both use the same connections and the protocol used is unique to WebSphere MQ Everyplace.

Connection definitions determine the links and protocols to be used for a particular connection. At each intermediate node any messages flowing through are passed to the queue manager at that point. The queue manager will handle the messages according to the resources it has. So a message may be placed on a queue which might be a local queue, a remote queue, or a store-and-forward queue. Messages placed on remote queues will continue their journey according to the type of remote queue. Synchronous remote queues will move the messages onward immediately. Asynchronous remote queues will store their messages before moving them.

Connections are not directly visible to applications or administrators and are established by the queue manager as required. Connections link queue managers together and their characteristics are changed by WebSphere MQ Everyplace, depending upon the information to be flowed. Transporters are the WebSphere MQ Everyplace components that exploit connections to provide queue level communication. Again, these are not visible to the application programmer or administrator.

When assured messaging is demanded, WebSphere MQ Everyplace delivers messages to the application once, and once-only. It achieves this by ensuring that a message has been successfully passed from one queue manager to another, and acknowledged, before deleting the copy at the transmitting end. In the event of a communications failure, if an acknowledgment has not been received, a message can be retransmitted, as once-only *delivery* does not imply once-only *transmission*. However, duplicates are not delivered.

Connection styles

WebSphere MQ Everyplace supports client/server operation. A *client* can initiate communication with a server. A *server* can respond only to the requests initiated by a client. The components involved are:

Listener
: Listens for incoming connection requests.

Queue manager
: Supports applications through the provision of messaging and queuing capabilities.

Table 3 on page 20 shows the relationship between these components and the connection style. The client/server connection style describes the situation where WebSphere MQ Everyplace can operate in either client or server mode. The servlet

option describes the case where WebSphere MQ Everyplace is configured as an HTTP servlet with the HTTP server itself responsible for listening for incoming connection requests.

Table 3. Connection styles

	Queue manager	Listener
Client	Yes	No
Client/server	Yes	Yes
Server	Yes	Yes
Servlet	Yes	No

WebSphere MQ Everyplace applications are not directly aware of the connection style used by the queue managers. However, the style is significant in that it affects what resources are available to the parties, which queue managers can connect with other queue managers, how much memory WebSphere MQ Everyplace uses, and which connections can concurrently exist.

Adapters

Adapters are used to map WebSphere MQ Everyplace to device interfaces. Channels exploit the protocol adapters to run over HTTP, native TCP/IP, UDP, and other protocols. Similarly, queues exploit field storage adapters to interface to a storage subsystem such as memory or the file system. Adapters provide a mechanism for WebSphere MQ Everyplace to extend its device support and allow version control. Unlike the WebSphere MQ Everyplace Java codebase, the C codebase uses the HTTP adapter only.

Dialup connection management

Dialup networking support for devices is handled by the device operating system. When WebSphere MQ Everyplace on a disconnected device attempts to use the network, for example because a message must be sent, and the network stack is not active, the operating system itself initiates remote access services (RAS). Typically this takes the form of a panel displayed to the user, offering a dialup connection profile. Until the connection is established, the operating system is in control. Consequently the device user must ensure that appropriate dialup connection profiles are available for the operating system to use. There is no explicit support for dialup networking in WebSphere MQ Everyplace.

Trace

Trace is enabled by running an independent program that performs tracing actions. Calls to trace for information, warning, and error situations are embedded within WebSphere MQ Everyplace. Applications can also call trace directly and, using the WebSphere MQ Everyplace Java codebase only, add new messages. As the interface that a trace handler must implement is published, solutions can implement this interface to collect WebSphere MQ Everyplace and application trace, interleave it, and direct the

output to where it can be collected. Several trace handlers are supplied as part of product code. Also, as most WebSphere MQ Everyplace exceptions are passed to the application for handling, the application exception handler can also route these to trace.

Event log

This section does not apply to the C codebase.

WebSphere MQ Everyplace provides event log mechanisms and interfaces that can be used to log status. For example, you can request a log entry when a queue manager starts. Logging is, by default, written to a file, but you can intercept this and direct it elsewhere. The WebSphere MQ Everyplace event log does not log message data and cannot be used to recover messages or queues.

Security

WebSphere MQ Everyplace provides an integrated set of security features enabling the protection of message data both when held locally and when being transferred.

WebSphere MQ Everyplace provides security under three different categories:

Local security
Protects message-related data at a local level

Queue-based security
Protects messages between the initiating queue manager and the target queue

Message-level security
Protects messages between the initiating and receiving WebSphere MQ Everyplace application

WebSphere MQ Everyplace local and message-level security are used internally by WebSphere MQ Everyplace, but are also made available to WebSphere MQ Everyplace applications. WebSphere MQ Everyplace queue-based security is an internal service.

The WebSphere MQ Everyplace security features of all three categories protect message data by use of an attribute, for example MQeAttribute. Depending on the category, the attribute is applied either externally or internally.

Each attribute can contain the following

Authenticator
Provides additional controls to prevent access to the local data by unauthorized users

Cryptor
Controls the strength of protection required

Compressor
Optimizes the size of the protected data

Key Controls access by requesting a password

Target entity name
Requests the target queue name

These elements are used differently, depending on the WebSphere MQ Everyplace security category, but in all cases the WebSphere MQ Everyplace security feature's protection is applied when the attribute attached to a message is invoked. Chapter 7, *Security*, of the WebSphere MQ Everyplace Application Programming Guide provides more information on the above elements and Chapter 6, *Configuration using administration messages*, of the WebSphere MQ Everyplace Configuration Guide describes how to write your own authenticator.

The registry

The registry is the main store for queue manager-related information. Each queue manager has at least one registry. Every queue manager uses the registry to hold its:

- Queue manager configuration data
- Queue definitions
- Remote queue definitions
- Connection definitions
- User data (including configuration-dependent security information)

Registry information is stored using an adapter, usually the MQeDiskFields adapter. Chapter 8, *Security*, of the WebSphere MQ Everyplace Application Programming Guide provides detailed information on the registry.

Private Registry and credentials

This section does not apply to the C codebase.

As every entity needs its own credentials to be authenticated, we need to know:

1. How to execute registration to get the credentials
2. Where to manage the credentials in a secure manner

The private registry enables the secure management of an entity's private credentials, and public registry manages set of public credentials. The private registry provides a base registry with secure or cryptographic token. For example, it can be a secure repository for public elements like mini-certificates, and private elements like private keys.

The private registry allows only authorized users to access the private elements. Normally, only the legitimate queue manager user can access the registry using a PIN. However, configuration options enable you to bypass this if you are not overly concerned with security issues.

The private registry provides support for services, for example digital signature and RSA decryption, in such a way that the private objects never leave the private registry. By providing a common interface, it hides the underlying device support, which currently

is restricted to the local file system. Chapter 8, *Security*, of the WebSphere MQ Everyplace Application Programming Guide provides detailed information on the private registry and credentials.

Auto-registration

WebSphere MQ Everyplace provides default services that support auto-registration. These services are automatically triggered when an authenticatable entity is configured, for example when a queue manager is started or when a new queue is defined. In both cases registration is triggered and new credentials are created and stored in the entity's private registry. Therefore, auto-registration provides a simple mechanism to establish credentials for message-level protection.

Auto-registration steps include:

1. Generating a new RSA key pair
2. Protecting and saving the private key in the private registry
3. Packaging the public key in a *new certificate* request to the default mini-certificate server

Assuming the mini-certificate server is configured and available, it returns the entity's new mini-certificate, along with its own. These servers and the protected private key are stored in the entity's private registry as its new credentials. Chapter 8, *Security*, of the WebSphere MQ Everyplace Application Programming Guide provides detailed information on auto-registration.

Public registry and certificate replication

WebSphere MQ Everyplace provides default services that enable WebSphere MQ Everyplace components to share mini-certificates. The WebSphere MQ Everyplace public registry provides a publicly accessible repository for mini-certificates. This is analogous to the personal telephone directory service on a mobile phone, the difference being that, instead of phone numbers, it is a set of mini-certificates of the authenticatable entities that are the most frequently contacted.

The public registry is not purely passive in its services. If accessed to provide a mini-certificate that it does not hold, and if configured with a valid home-server component, the public registry automatically attempts to fetch the requested mini-certificate from the public registry of the home server. These services can be used to provide an intelligent automated mini-certificate replication service that makes the right mini-certificate available at the right time.

Application use of registry services

The WebSphere MQ Everyplace queue manager exploits the advantages of using private and public registry services, but access to these services is not restricted. WebSphere MQ Everyplace solutions can define and manage their own entities, such as users. You can then use private registry services to auto-register and manage the credentials of the new entities, and public registry services to make the public credentials available where needed. Chapter 8, *Security*, of the WebSphere MQ Everyplace Application Programming Guide provides detailed information on how to use registry services.

Default mini-certificate issuance service

The ES03 WebSphere MQ Everyplace SupportPac, ″Websphere MQ Everyplace WTLS Mini-Certificate Server″ is available as a separate free download from http://www.ibm.com/software/mqseries/txppacs/. This software package provides a certificate issuance service for WTLS certificates. You can configure queue manager and queue entities on this certificate issuance server to provide a default mini-certificate issuance service that satisfies private-registry auto-registration requests with the issuance of WTLS certificates. You can use the WebSphere MQ Everyplace certificate issuance service to set up and manage a mini-certificate issuance service to issue mini-certificates to a carefully controlled set of entity names. The characteristics of this issuance service are:

- Management of the set of registered authenticatable entities
- Mini-certificate issuance
- WAP WTLS mini-certificate repository management

Chapter 8, *Security*, of the WebSphere MQ Everyplace Application Programming Guide also provides detailed information on issuing mini-certificates. Also, refer to the documentation included in the ES03 SupportPac for more details of how to install and use the WTLS digital certificate issuance service for WebSphere MQ Everyplace.

The security interface

An optional interface is provided that can be implemented by a custom security manager. The methods allow the security manager to authorize or reject requests associated with:

- Adding and removing class aliases
- Defining adapters
- Mapping file descriptors
- Processing connection commands

Customizing Rules

Rules allow users to customize the behavior of some main WebSphere MQ Everyplace components. WebSphere MQ Everyplace provides default rules where necessary, but you can replace these with application or installation-specific rules to meet customer requirements. The rule types supported differ in how they are triggered and in what they can do. Rules contain logic and can therefore perform a wide range of functions.

Attribute rules

Attribute rules apply to the Java codebase only. This rule class is given control whenever a change of state is attempted, for example, a change of:

- Authenticator
- Compressor
- Cryptor

The rule would normally allow or disallow the change.

WebSphere MQ bridge rules

WebSphere MQ bridge rules apply to the Java codebase only. These rule classes are given control when the WebSphere MQ Everyplace to WebSphere MQ bridge code has a change of state. There is a separate bridge rule class to determine each of the following:

- What to do with a message when a listener cannot deliver it to WebSphere MQ Everyplace, when it is coming from WebSphere MQ, for example, because the message is too big or the queue does not exist
- What state the bridge-administered queues should start in once the server is instantiated
- What to do when the bridge finds something wrong with the WebSphere MQ Sync queue, that is the persistent store used for crash recovery (the default rule displays the problem only)
- How to convert a WebSphere MQ Everyplace message to a WebSphere MQ message, and vice-versa using transformers

Chapter 2, *Rules*, of the WebSphere MQ Everyplace System Programming Guide and Chapter 7, *Interoperability with other messaging systems*, of the WebSphere MQ Everyplace Application Programming Guide provide detailed information on WebSphere MQ bridge rules.

Queue rules

This rule class is invoked at key points in the lifestyle of a queue, for example when:

- A message is added to a queue, for example, to see if a threshold is exceeded, that is number of messages, size of message, or invalid priority.
- A queue is opened or closed.
- A queue is removed from a queue manager. This does not apply to the native C codebase.
- A message on a queue has exceeded either the queue's or its own expiry interval.

Queue manager rules

This rule class invoked at key points in the lifecycle of a queue manager, for example when:

- A queue manager is opened, for example, to start a background timer thread running to allow timed actions to occur.
- A queue manager is closed, for example, to terminate the background timer thread.
- The transmission of the queue manager's pending messages is triggered.

Classes

This section does not apply to the WebSphere MQ Everyplace native C codebase.

WebSphere MQ Everyplace provides a choice of classes for certain functions that allow you to customize WebSphere MQ Everyplace behavior to meet specific application requirements. In some cases the interfaces to classes are documented so that

additional alternatives can be developed. Table 4 on page 27 summarizes the possibilities. Classes can be identified either explicitly or through the use of alias names.

Note: Some of the classes are not provided in the C Bindings API. See *WebSphere MQ Everyplace Java Programming Reference* and *WebSphere MQ Everyplace C Programming Reference* on the product CD for definitive lists of the supported classes.

Table 4. Class options

Class	Alternatives supplied	Interfaces documented
Administration	No	Yes
Authenticators	Yes	No
Communications adapter	Yes	Yes
Communications style	Yes	No
Compressors	Yes	No
Cryptors	Yes	No
Event log	Sample provided	Yes
Messages	No	Yes
Queue storage	Yes	No
Rules	Default classes provided	Yes
Trace	Samples provided	Yes

Application loading

This section does not apply to the C codebase.

When a WebSphere MQ Everyplace queue manager is loaded, the initiating application must load any other applications into the JVM. Standard Java facilities can be used for this, or you can use the class loader included as part of WebSphere MQ Everyplace. Therefore, multiple applications can run against a single queue manager in the same JVM. Alternatively, you can use multiple JVM, but each requires its own queue manager and each of these must have a unique name.

Chapter 4. The WebSphere MQ Everyplace API and WebSphere MQ

This chapter describes how to connect to a WebSphere MQ server and the *WebSphere MQ Everyplace Application Programming Interface (API)*. However, before reading this chapter, we recommend that you first install WebSphere MQ Everyplace. For information on server platforms that support WebSphere MQ Everyplace, storage requirements, and installing WebSphere MQ Everyplace, refer to Chapter 1, *Installation procedures*, of the WebSphere MQ Everyplace Read Me First.

This chapter describes:

- WebSphere MQ Everyplace and WebSphere MQ networks
- Programming Interfaces

WebSphere MQ Everyplace and WebSphere MQ networks

Although a WebSphere MQ Everyplace network can exist standalone, without the need for a WebSphere MQ server or network, in practice WebSphere MQ Everyplace is often used to complement an existing WebSphere MQ installation. This extends WebSphere MQ's reach to new platforms and devices, and provides advanced capabilities, such as queue or message based security and synchronous messaging. From a WebSphere MQ Everyplace application perspective, WebSphere MQ queues and queue managers act as additional remote queues and queue managers. However, a number of functional restrictions exist because these queues are not accessed directly through WebSphere MQ Everyplace connections and a WebSphere MQ Everyplace queue manager, but require the involvement of an WebSphere MQ Everyplace gateway.

The gateway can send messages to multiple WebSphere MQ queue managers either directly or indirectly, through WebSphere MQ client channels. If the connection is indirect, the messages pass through WebSphere MQ client channels to an intermediate WebSphere MQ queue manager and then onwards through WebSphere MQ message channels to the target queue manager. Chapter 7, *Interoperability with other messaging systems*, of the WebSphere MQ Everyplace Application Programming Guide provides more information on the gateway and message routing between WebSphere MQ Everyplace and WebSphere MQ.

Interface to WebSphere MQ

This section does not apply to the C codebase.

WebSphere MQ Everyplace supports the *WebSphere MQ bridge*, which acts as an interface between WebSphere MQ Everyplace and WebSphere MQ networks. This bridge uses the WebSphere MQ Java client to interface to one or more WebSphere MQ queue managers, thereby allowing messages to flow from WebSphere MQ Everyplace to WebSphere MQ and vice versa. In the current version of WebSphere MQ Everyplace one such bridge is recommended per server, and each is associated with multiple *WebSphere MQ queue manager proxies* (definitions of WebSphere MQ queue managers). A queue manager proxy definition is required for each WebSphere MQ queue manager that communicates with WebSphere MQ Everyplace. Each of these

definitions can have one or more associated *client connection services*, where each represents a connection to a single WebSphere MQ queue manager. Each of these may use a different WebSphere MQ server connection to the queue manager, and optionally a different set of properties such as user exits or ports.

Chapter 7, Interoperability with other messaging systems, of the WebSphere MQ Everyplace Application Programming Guide describes *WebSphere MQ bridge* properties in detail, how messages flow between WebSphere MQ Everyplace and WebSphere MQ using the *WebSphere MQ bridge*, and how to customize the bridge for your specific purposes.

Message conversion

WebSphere MQ Everyplace messages destined for WebSphere MQ pass through the bridge and are converted into a WebSphere MQ format, using either a default transformer or one specific to the target queue. A custom transformer offers much flexibility, for example it is good practice to use a subclass of the WebSphere MQ Everyplace message class to represent messages of a particular type over the WebSphere MQ Everyplace network. On the gateway a transformer can convert the message into a WebSphere MQ format using appropriate mapping between fields and WebSphere MQ values and adding specific data to represent the significance of the subclass.

The default transformer from WebSphere MQ Everyplace to WebSphere MQ cannot take advantage of subclass information but has been designed to be useful in a wide range of situations. It has the following characteristics:

- **Message flow from WebSphere MQ Everyplace to WebSphere MQ:**

  The default transformer from WebSphere MQ Everyplace to WebSphere MQ works in conjunction with the *MQeMQMsgObject* class. This class is a representation of all the fields you could find in a WebSphere MQ message header. Using the MQeMQMsgObject, your application can set values using set() methods. Therefore, when an MQeMQMsgObject, or an object derived from it, is passed through the default WebSphere MQ Everyplace transformer, that is the MQeBaseTransformer, the MQeBaseTransformer gets the values from inside the MQeMSMsgObject, and sets the corresponding values in the WebSphere MQ message, for example, the priority value is copied over to the WebSphere MQ message.

  If the message being passed is not an MQeMQMsgObject, and is not derived from the MQeMQMsgObject class, the whole WebSphere MQ Everyplace message is copied into the body of the WebSphere MQ message. This is referred to as *funneling*. The message format field in the WebSphere MQ message header is set to indicate that the WebSphere MQ message holds a message in WebSphere MQ Everyplace ″funneled″ format.

- **WebSphere MQ to WebSphere MQ Everyplace message flow:**

  WebSphere MQ messages for WebSphere MQ Everyplace are handled similarly to those travelling in the other direction. The default transformer inspects the message type field of the WebSphere MQ header and acts accordingly.

If the WebSphere MQ header indicates a ″funneled″ WebSphere MQ Everyplace message, then the WebSphere MQ message body is reconstituted as the original WebSphere MQ Everyplace message that is then posted to the WebSphere MQ Everyplace network.

If the message is not a ″funneled″ WebSphere MQ Everyplace message, then the WebSphere MQ message header content is extracted, and placed into an MQeMQMsgObject. The WebSphere MQ message body is treated as a simple byte field, and is also placed into the MQeMQMsgObject. The MQeMQMsgObject is then posted to the WebSphere MQ Everyplace network.

This MQeMQMsgObject class and the default transformer behavior mean that:

- A WebSphere MQ Everyplace message can travel across a WebSphere MQ network to a WebSphere MQ Everyplace network without change.
- A WebSphere MQ message can travel across a WebSphere MQ Everyplace network to a WebSphere MQ network without change.
- A WebSphere MQ Everyplace application can drive any existing WebSphere MQ application without the WebSphere MQ application being changed.

Function

WebSphere MQ remote queues are enabled for synchronous WebSphere MQ Everyplace put messaging operations, from a WebSphere MQ Everyplace queue manager. All other messaging operations must be asynchronous.

WebSphere MQ Everyplace administration messages cannot be sent to a WebSphere MQ queue manager. The administration queue does not exist there and the administration message format differs from that used by WebSphere MQ.

Compatibility

A WebSphere MQ Everyplace network can exist independently of WebSphere MQ, but in many situations the two products together are needed to meet the application requirements. WebSphere MQ Everyplace can integrate into an existing WebSphere MQ network with compatibility including the aspects summarized below:

Addressing and naming:

- Identical addressing semantics using a queue manager or queue address
- Common use of an ASCII name space

Applications:

WebSphere MQ Everyplace is able to support existing WebSphere MQ applications without application change.

Connections:

The WebSphere MQ Everyplace gateway uses WebSphere MQ client channels.

Message interchange and content:

- Interchange of messages between WebSphere MQ Everyplace and WebSphere MQ

- Message network invisibility (messages from either WebSphere MQ Everyplace or WebSphere MQ can cross the other network without change)
- Mutual support for identified fields in the WebSphere MQ message header
- Once and once-only assured message delivery

WebSphere MQ Everyplace Version 2.0 does not support all the functions of WebSphere MQ. Apart from environmental, operating system and communication considerations, some of the more significant differences:

- No clustering support
- No distribution list support
- No grouped or segmented messages
- No load balancing or warm standby capabilities
- No reference message
- No report options
- No shared queue support
- No triggering
- No unit of work support, no XA-coordination
- Different scalability and performance characteristics

However, within WebSphere MQ Everyplace many application tasks can be achieved through alternative means using WebSphere MQ Everyplace features, or through the exploitation of sub-classing, the replacement of the supplied classes or the exploitation of the rules, interfaces, and other customization features built into the product.

Assured delivery

Although both WebSphere MQ Everyplace and WebSphere MQ offer assured delivery, they each provide for different levels of assurance. When a message is travelling from WebSphere MQ Everyplace to WebSphere MQ, the message transfer is only assured if the combination of *putMessage* and *confirmPutMessage* is used. Chapter 6, *Message Delivery*, of the WebSphere MQ Everyplace Application Programming Guide, provides detailed information on transferring messages. When a message is travelling from WebSphere MQ to WebSphere MQ Everyplace, the transfer is assured only if the WebSphere MQ message is defined as persistent.

Programming Interfaces

The WebSphere MQ Everyplace Application Programming Interface (API) is the programming interface to WebSphere MQ Everyplace. Two languages are supported, Java and C.

The Java version provides access to all WebSphere MQ Everyplace functions. The detailed classes, methods, and procedures are described in the *WebSphere MQ Everyplace Java Programming Reference* on the product CD. Examples of WebSphere MQ Everyplace programming are given in the *WebSphere MQ Everyplace Application Programming Guide*.

There are three versions of the C support:

The Native C codebase provides access to a major subset of WebSphere MQ Everyplace functions. As the C codebase is a device queue manager only it:

- Does not support store-and-forward queues or bridge queues
- Supports the HTTP adapter only
- Supports the RLE compressor only
- Supports the RC4 cryptor only
- Supports the *MAttribute* and local security features only

The detailed methods and procedures are described in the *WebSphere MQ Everyplace C Programming Reference* on the product CD. Examples of programming WebSphere MQ Everyplace for the C bindings are given in the *WebSphere MQ Everyplace Application Programming Guide*.

The C Bindings provide access to a major subset of WebSphere MQ Everyplace functions. The detailed methods and procedures are described in the *WebSphere MQ Everyplace C Programming Reference* on the product CD. Examples of programming WebSphere MQ Everyplace for the C bindings are given in the *WebSphere MQ Everyplace C Bindings Programming Guide*.

The C support for Palm provides access for a subset of the WebSphere MQ Everyplace function for use on Palm devices. Details of these classes and procedures, together with programming guidance, are provided in *WebSphere MQ Everyplace C Programming Guide for Palm OS*.

Chapter 5. Getting started with WebSphere MQ Everyplace

WebSphere MQ Everyplace is a family of products that collectively provide the tools needed to develop, deploy, and manage WebSphere MQ Everyplace messaging and queuing solutions. The family comprises:

1. The *WebSphere MQ Everyplace licensed product*, available on physical media from IBM or as a Web download from:
   http://www-3.ibm.com/software/
   The licensed product includes:
   - WebSphere MQ Everyplace Java classes
   - Helper classes
   - WebSphere MQ Everyplace C Bindings files and native C codebase
   - Application source code examples
   - Utilities
   - Reference manuals
   - License information

   The physical Program Product also includes entitlement to use the product for non-development use on certain platforms. Further capacity units need to be purchased for use on larger machines, or with the WebSphere MQ bridge.
2. WebSphere MQ Everyplace SupportPacs, available as Web downloads from:
   http://www-3.ibm.com/software/ or
   http://www.ibm.com/software/mqseries/everyplace. These are essential supplements to the licensed product and include for example:

   EAP1: WebSphere MQ Everyplace - Device code for the Palm OS
   C programming language support for WebSphere MQ Everyplace Version 1.0.1 application development on the Palm OS. This code is also included on the product CD inside the file eap1.zip.

   EP01: WebSphere MQ Everyplace - Performance Report
   This analyses WebSphere MQ Everyplace performance on a variety of client platforms.

   ES01: WebSphere MQ Everyplace - Administration Tool (MQeExplorer v1.0)
   This s a generic tool for all Java platforms enabling easy graphical administration of WebSphere MQ Everyplace queue managers.

   ES02: WebSphere MQ Everyplace - Explorer (MQe_Explorer v1.2)
   This is a WebSphere MQ Everyplace administration tool developed exclusively to support the Microsoft Windows range of operating systems.

   ED03: WebSphere MQ Everyplace - Designer
   This is a tool for all Java platforms that aids in designing, validating, and administering WebSphere MQ Everyplace networks.

The management tools in the WebSphere MQ Everyplace SupportPacs play an important role in all phases of application development and rollout. They are more

sophisticated than the utilities included with the licensed product and are an essential aid to getting started, configuring, inspecting pilot networks, and managing production systems.

Using WebSphere MQ Everyplace

Given the wide range of uses for WebSphere MQ Everyplace, the product is not installed, configured, and deployed in the same way as other members of the WebSphere MQ family. There are three phases in the adoption of WebSphere MQ Everyplace:

1. Development and prototyping phase

WebSphere MQ Everyplace is available for installation and use without charge, subject to the conditions of the WebSphere MQ Everyplace development license. WebSphere MQ Everyplace applications are developed, using the functions of the Java classes and C API. These applications can be packaged in a variety of ways.

- In Java, you can set up a WebSphere MQ Everyplace queue manager as a daemon with one or more applications launched into the same Java virtual machine (JVM) and sharing a common queue manager.
- In C, to develop applications using the WebSphere MQ Everyplace Development Kit, you need Microsoft eMbedded Visual Tools 3.0 and an SDK for your chosen platform.
- In Java, the application embeds the required WebSphere MQ Everyplace classes such that the application runs on machines where WebSphere MQ Everyplace has not been installed, launching its own queue manager into its own JVM.
- In Java, the application uses the WebSphere MQ Everyplace java classes and C APIs that exist on the target machine.

Support from IBM is not included with the development license. However, support during application development and beyond is provided with the deployment license (see below).

2. Deployment phase

The deployment phase refers to how you use the developed applications and, therefore, under the terms of the WebSphere MQ Everyplace license, capacity units are required to use the product. The java classes and C API can only be distributed with the application with agreement from IBM, or where the users already have entitlement to use them. Otherwise, in java, users must customize the necessary classes themselves and, in C, copy the WebSphere MQ Everyplace to the device.

3. Management phase

Subsequently, when WebSphere MQ Everyplace queue managers are active within a network, tools are needed to inspect and manage them. Support for WebSphere MQ Everyplace is provided under the terms of the International Program License Agreement.

This adoption life cycle explains the variation in level of support with platforms. For the WebSphere MQ Everyplace with capacity units, and Category 3 SupportPacs, IBM distinguishes between:

- Platforms where installation and application development is supported:
  - Problem reports on install, application development, and use are accepted
- Platforms where the application deployment is permitted but not directly supported:
  - problem reports might be required to be reproduced on a supported platform
- Platforms where application deployment is supported:
  - Problem reports resulting from application deployment are accepted

Gaining experience

There are many ways to get started with WebSphere MQ Everyplace. Getting a queue manager up and running, followed by setting up a simple WebSphere MQ Everyplace network, is a productive way to become familiar with the product and its concepts. Writing a simple application is sound preparation for in-depth study of the product details. In the early stages it is generally not helpful to examine other members of the WebSphere MQ family. Later, when the bridge functionality is of interest, this understanding becomes essential.

With this strategy in mind, new users are recommended to understand the essentials of the concepts presented in this book. If you have access to a machine running a Windows operating system, download the MQe_Explorer, SupportPac ES02 and follow the instructions given to get started with WebSphere MQ Everyplace. You do not have to install the licensed product beforehand, but if you do not, you are restricted by the terms of the license.

Appendix. Notices

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

IBM may have patents or pending patent applications covering subject matter described in this document. The furnishing of this document does not 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.

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

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

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

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 Program License Agreement, or any equivalent agreement between us.

Trademarks

The following terms are trademarks of International Business machines Corporation in the United States, or other countries, or both.

AIX Everyplace IBM iSeries MQSeries WebSphere z/OS zSeries

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

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

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

Glossary

This glossary describes terms used in this book and words used with other than their everyday meaning. In some cases, a definition might not be the only one applicable to a term, but it gives the particular sense in which the word is used in this book.

If you do not find the term you are looking for, see the index or the *IBM Dictionary of Computing*, New York:. McGraw-Hill, 1994.

application programming interface (API). An application programming interface consists of the functions and variables that programmers are allowed to use in their applications.

asynchronous messaging. A method of communication between programs in which programs place messages on message queues. With asynchronous messaging, the sending program proceeds with its own processing without waiting for a reply to its message. Contrast with *synchronous messaging*.

authenticator. A program that verifies the senders and receivers of messages.

bridge. A WebSphere MQ Everyplace object that allows messages to flow between WebSphere MQ Everyplace and other messaging systems, including WebSphere MQ.

channel. See *dynamic channel* and *MQI channel*.

channel manager. A WebSphere MQ Everyplace object that supports logical multiple concurrent communication pipes between end points.

class. An encapsulated collection of data and methods to operate on the data. A class may be instantiated to produce an object that is an instance of the class.

client. In WebSphere MQ, a client is a run-time component that provides access to queuing services on a server for local user applications.

compressor. A program that compacts a message to reduce the volume of data to be transmitted.

connection. Links WebSphere MQ Everyplace devices and transfers synchronous and asynchronous messages and responses in a bidirectional manner.

cryptor. A program that encrypts a message to provide security during transmission.

encapsulation. An object-oriented programming technique that makes an object's data private or protected and allows programmers to access and manipulate the data only through method calls.

gateway. In WebSphere MQ Everyplace, a computer running the WebSphere MQ Everyplace code including a channel manager.

Hypertext Markup Language (HTML). A language used to define information that is to be displayed on the World Wide Web.

instance. An object. When a class is instantiated to produce an object, the object is an instance of the class.

interface. A class that contains only abstract methods and no instance variables. An interface provides a common set of methods that can be implemented by subclasses of a number of different classes.

Internet. A cooperative public network of shared information. Physically, the Internet uses a subset of the total resources of all the currently existing public telecommunication networks. Technically, what distinguishes the Internet as a cooperative public network is its use of a set of protocols called TCP/IP (Transport Control Protocol/Internet Protocol).

Java Development Kit (JDK). A package of software distributed by Sun Microsystems for Java developers. It includes the Java interpreter, Java classes and Java development tools: compiler,

debugger, disassembler, appletviewer, stub file generator, and documentation generator.

Java Naming and Directory Service (JNDI). An API specified in the Java programming language. It provides naming and directory functions to applications written in the Java programming language.

Lightweight Directory Access Protocol (LDAP). A client/server protocol for accessing a directory service.

message. In message queuing applications, a communication sent between programs.

message queue. See *queue*.

message queuing. A programming technique in which each program within an application communicates with the other programs by putting messages on queues.

method. The object-oriented programming term for a function or procedure.

MQI channel. Connects a WebSphere MQ client to a queue manager on a server system and transfers MQI calls and responses in a bidirectional manner.

WebSphere MQ. WebSphere MQ is a family of IBM licensed programs that provide message queuing services.

object. (1) In Java, an object is an instance of a class. A class models a group of things; an object models a particular member of that group. (2) In WebSphere MQ, an object is a queue manager, a queue, or a channel.

package. A package in Java is a way of giving a piece of Java code access to a specific set of classes. Java code that is part of a particular package has access to all the classes in the package and to all non-private methods and fields in the classes.

personal digital aistant (PDA). A pocket sized personal computer.

private. A private field is not visible outside its own class.

protected. A protected field is visible only within its own class, within a subclass, or within packages of which the class is a part.

public. A public class or interface is visible everywhere. A public method or variable is visible everywhere that its class is visible.

queue. A queue is a WebSphere MQ object. Message queueing applications can put messages on, and get messages from, a queue.

queue manager. A queue manager is a system program that provides message queuing services to applications.

registry. Stores the queue manager configuration information.

server. (1) An WebSphere MQ Everyplace server is a device that has a WebSphere MQ Everyplace channel manager configured. (2) A WebSphere MQ server is a queue manager that provides message queuing services to client applications running on a remote workstation. (3) More generally, a server is a program that responds to requests for information in the particular two-program information flow model of client/server. (3) The computer on which a server program runs.

servlet. A Java program which is designed to run only on a Web server.

subclass. A subclass is a class that extends another. The subclass inherits the public and protected methods and variables of its superclass.

superclass. A superclass is a class that is extended by some other class. The superclass's public and protected methods and variables are available to the subclass.

synchronous messaging. A method of communicating between programs in which programs place messages on message queues. With synchronous messaging, the sending program waits for a reply to its message before

resuming its own processing. Contrast with *asynchronous messaging*.

Transmission Control Protocol/Internet Protocol (TCP/IP). A set of communication protocols that support peer-to-peer connectivity functions for both local and wide area networks.

transformer. A piece of code that performs data or message reformatting.

Web. See World Wide Web.

Web browser. A program that formats and displays information that is distributed on the World Wide Web.

World Wide Web (Web). The World Wide Web is an Internet service, based on a common set of protocols, which allows a particularly configured server computer to distribute documents across the Internet in a standard way.

Bibliography

Related publications:

- *WebSphere MQ Everyplace Read Me First*, GC34-6276-00
- *WebSphere MQ Everyplace Application Programming Guide*, SC34-6278-00
- *WebSphere MQ Everyplace System Programming Guide*, SC34-6274-00
- *WebSphere MQ Everyplace C Bindings Programming Guide*, SC34-6280-00
- *WebSphere MQ Everyplace Java Programming Reference*
- *WebSphere MQ Everyplace C Programming Reference*
- *WebSphere MQ Everyplace C Programming Guide for Palm OS*, SC34-6281-00
- *WebSphere MQ An Introduction to Messaging and Queuing*, GC33-0805-01
- *MQSeries for Windows NT V5R1 Quick Beginnings **, GC34-5389

Note: * Or the appropriate Quick Beginnings book for your platform.

Index

R

S

T

W

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 44–1962–842327
  - From within the U.K., use 01962–842327
- 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.

SC34-6277-00