Unit 1. A review of WebSphere MQ

What this unit is about

This unit provides an introduction to WebSphere MQ and its products. It forms the basis for the remainder of the course.

What you should be able to do

After completing this unit, you should be able to:

- Describe the features and benefits of WebSphere MQ
- Identify the level of function in each WebSphere MQ queue manager
- Classify the application models that WebSphere MQ can support
- Find further information about specific aspects of WebSphere MQ

How you will check your progress

- Checkpoint questions
- Instructor questions

References

SC34-6941	WebSphere MQ Script (MQSC) Command Reference
SC34-6928	WebSphere MQ System Administration Guide

Unit objectives

After completing this unit, you should be able to:

- Describe the features and benefits of WebSphere MQ
- Identify the level of function in each WebSphere MQ queue manager
- Classify the application models that WebSphere MQ can support
- Find further information about specific aspects of WebSphere MQ

Figure 1-1. Unit objectives WM203 / VM2032.0

Notes:

WebSphere MQ functional overview

- Common application programming interface
- Assured message delivery
- Time-independent processing
- Application parallelism
- Faster application development

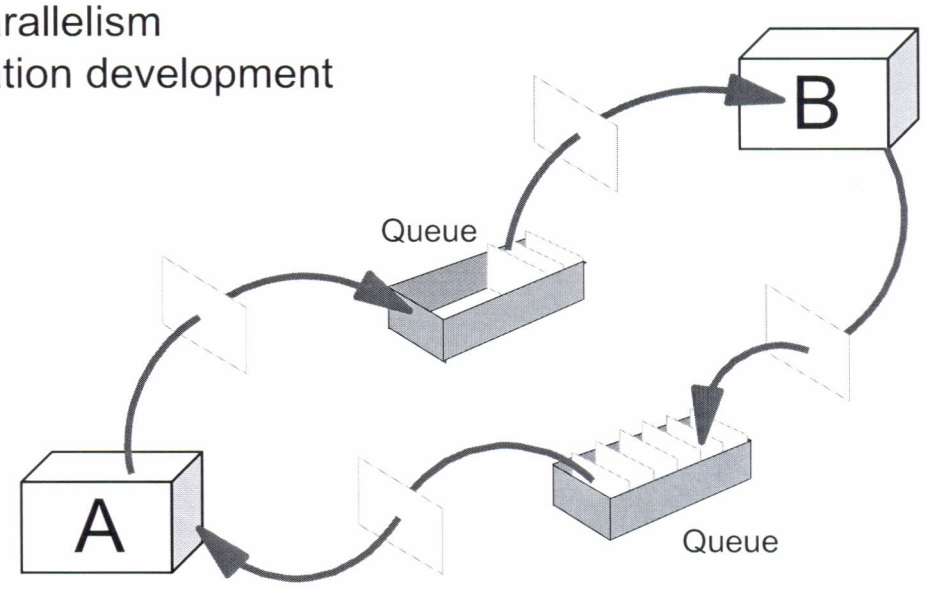


Figure 1-2. WebSphere MQ functional overview WM203 / VM2032.0

Notes:

WebSphere MQ provides program-to-program communication using messages and queues. The communicating applications can be on the same system, or distributed across a network of IBM and non-IBM systems.

The five major benefits of WebSphere MQ are as follows:

- There is a common application programming interface, the MQI, that is consistent across all the supported platforms.
- WebSphere MQ can transfer data with assured delivery; messages do not get lost, even in the event of a system failure. There is no duplicate delivery.
- Communicating applications do not have to be active at the same time.
- An application is divided into discrete functional modules which communicate with each other with messages. In this way, the modules can run on different systems, be scheduled at different times, or they can act in parallel.
- Application development is faster because the developer is shielded from the complexities of the network.

Additional information

- WebSphere MQ publications
  - Cross-platform
    - For information that is common
    - For planning and implementing a WebSphere MQ network
  - Platform-specific

- http://www.ibm.com/software/integration/wmq/
  - Latest news
  - References
  - Beta code
  - SupportPacs

Figure 1-3. Additional information

Notes:

The WebSphere MQ publications are listed in the bibliography at the back of these course notes. Some publications describe function that relates to two or more queue managers, the so called cross-platform publications. Other publications are platform-specific.

Discover WebSphere MQ on the World Wide Web. The web address for the WebSphere MQ home page is:

`http://www.ibm.com/software/integration/wmq`

Message and queue

- A message is:
  - A unit of information
  - A request for a service
  - A reply
  - A report
  - An announcement
- A queue is:
  - A safe place to store messages
  - Lined up for servicing
  - Staged for delivery

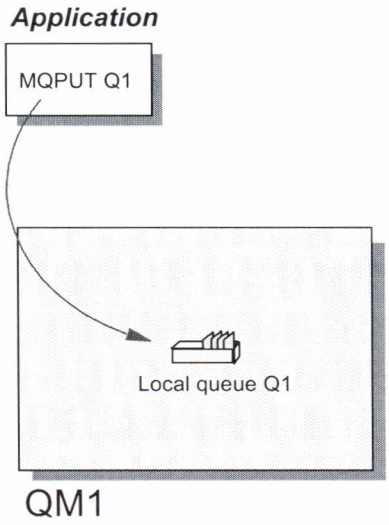


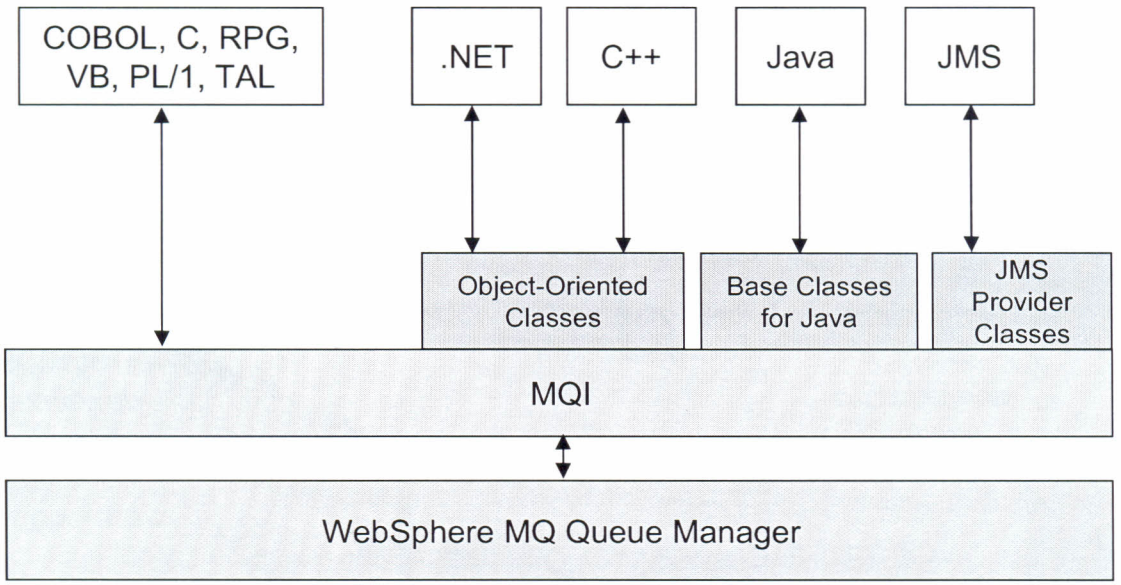
Figure 1-4. Message and queue

Notes:

A *message* is any information that one application wants to communicate to another. A message can convey a request for a service, or it can be a reply to such a request. It might also report on the progress of another message; to confirm its arrival or report on an error, for example. A message can also carry information for which no reply is expected.

A *queue* is a place to store messages until they can be processed. The time a message has to wait in order to be retrieved and processed can be short. Alternatively, it can be a long time if it has to wait for the receiving application to be started. Either way, the ability to store a message safely is an important characteristic of a queue.

Queue manager



- Queue manager owns and manages queues
- Application programming interfaces enable applications to access queues and messages
  - Message Queue Interface (MQI)
  - Java Messaging Service (JMS)

Figure 1-5. Queue manager WM203 / VM2032.0

Notes:

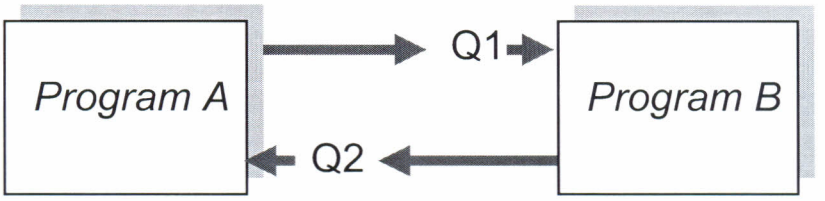
The component of WebSphere MQ software which owns and manages queues is called a *queue manager.*

A queue manager also provides a family of application programming interfaces.

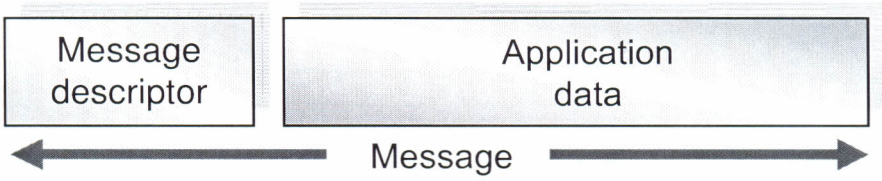
- The **Message Queue Interface (MQI)** enables an application to access its queues and the messages they contain. The MQI is a simple application programming interface which is consistent across all platforms supported by WebSphere MQ. The MQI effectively protects applications from having to know how a queue manager physically manages messages and queues. The MQI allows full access to WebSphere MQ messaging support.
- The **Java Message Service (JMS)** is a specification of a portable API for asynchronous messaging. JMS is an object-oriented Java API with a set of generic messaging objects for programmers to write event-based messaging applications. JMS supports both request/reply and publish/subscribe models as separate object models.

Both of the APIs can interoperate.

Message descriptor



- Each message has a message descriptor
- Application determines message content



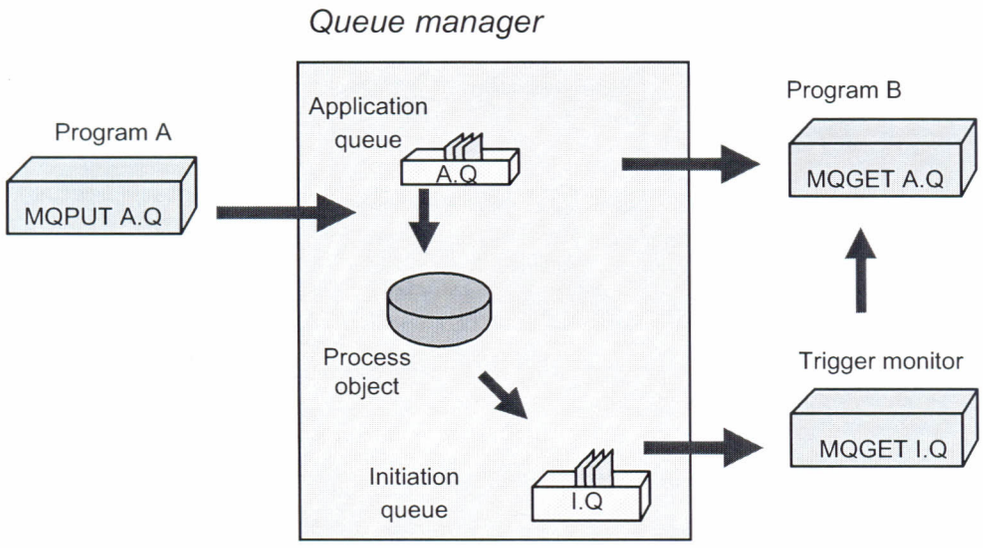
Notes:

A message consists of two parts:

- Message descriptor
- Application data

The *message descriptor* contains information about the message. The sending application supplies both the message descriptor and the application data when it puts a message on a queue. Both the message descriptor and the application data are returned to the application which gets the message from the queue. Some of the fields in the message descriptor are set by the application which puts the message on a queue; others are set by the queue manager on behalf of the application.

Triggering



- Triggering allows
  - Instantiation as required
  - Conservation of system resources
  - Automation of flow

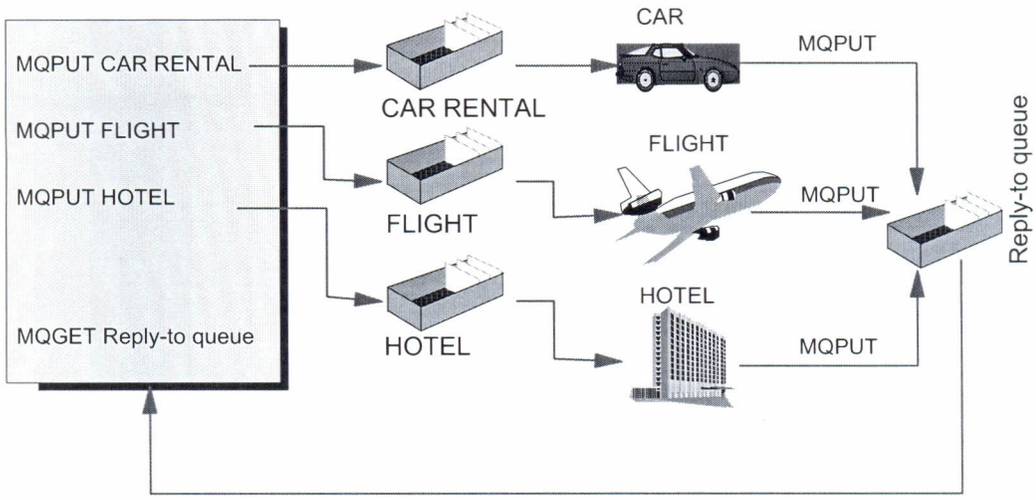
Figure 1-7. Triggering WM203 / VM2032.0

Notes:

WebSphere MQ provides an enhancement to the implementation of time-independent processing, *triggering*. The arrival of a message on an application queue might indicate that it is an appropriate time for another application to be started in order to process the messages on the application queue. When the right conditions are detected by the queue manager, it is the triggering facility which starts the application to service the application queue.

This course revisits triggering in some depth later.

Parallel processing



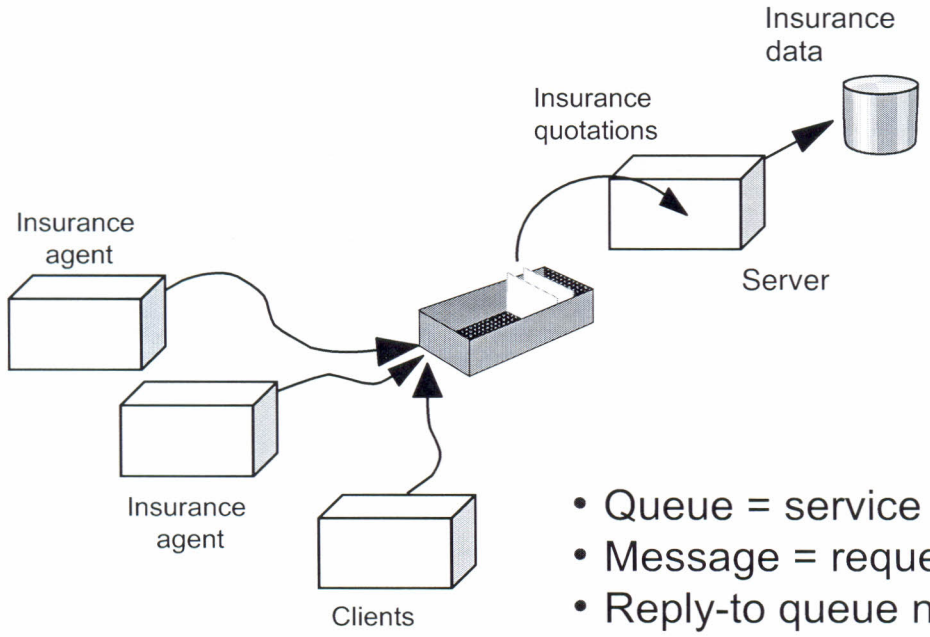
- Requests not serialized
- Shorter elapsed time
- Consolidate replies
- Possibly selective MQGET?
- Incomplete set of replies?
- Different process to handle replies?

Notes:

This model allows several requests to be sent by an application without the application having to wait for a reply to one request before sending the next. All the requests can then be processed in parallel.

The application can process the replies when they have all been received, and produce a consolidated answer. The program logic might also specify what to do when only a partial set of replies is received within a given period.

Requesting and responding applications



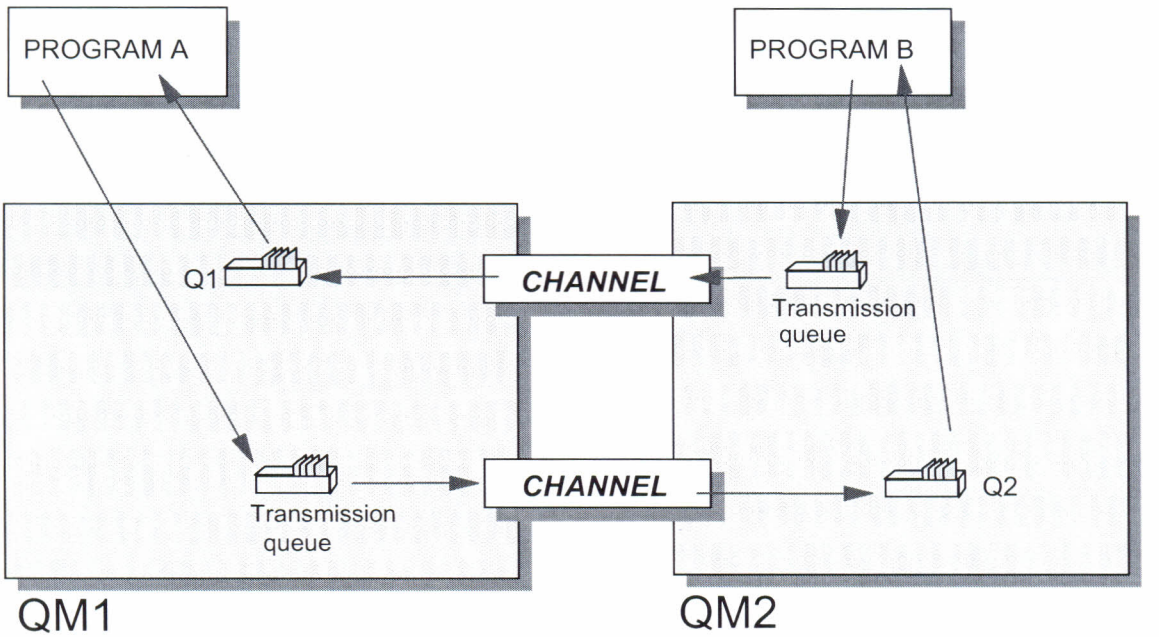
- Queue = service
- Message = request
- Reply-to queue name in message descriptor
- Multiple instances of server possible

Figure 1-9. Requesting and responding applications WM203 / VM2032.0

Notes:

The server application Insurance quotations, can handle requests from multiple client applications. The message descriptor identifies the appropriate reply-to queue for each request.

Transmission queues and channels



- Transmission queue stores message first
- Application not stopped if link is inactive

Figure 1-10. Transmission queues and channels WM203 / VM2032.0

Notes:

Assured delivery is another benefit of WebSphere MQ. It is the result of the protocol used when one queue manager transmits a message to another queue manager. As far as the applications are concerned, this process of transmitting messages is performed asynchronously and transparently. Furthermore, the protocol ensures that no message is lost or delivered to its destination queue more than once.

Transmission queue header

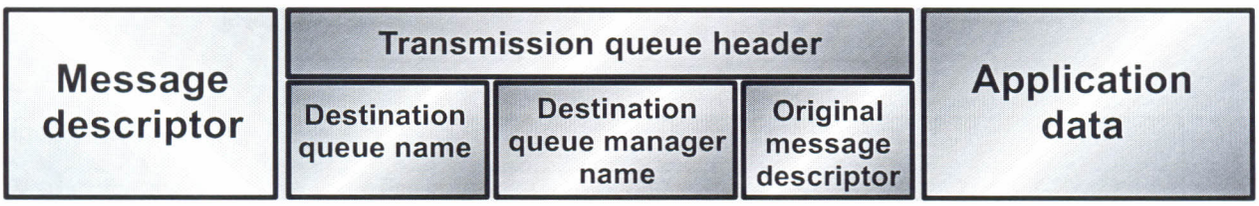


Figure 1-11. Transmission queue header WM203 / VM2032.0

Notes:

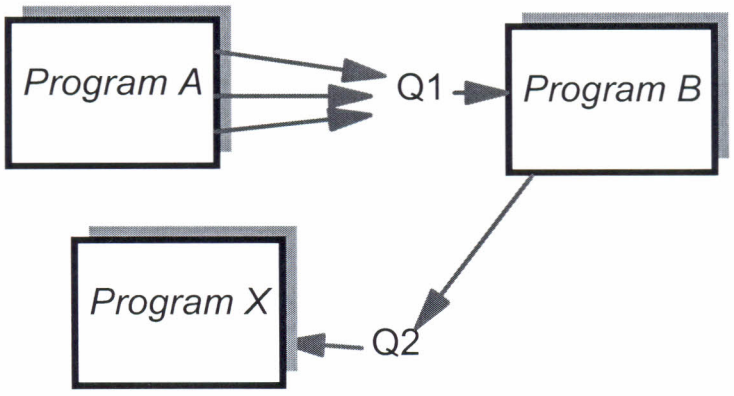
When a queue manager puts a message on a transmission queue, it places a *transmission queue header* in front of the application data. The transmission queue header contains, among other things, the following information:

- The name of the destination queue
- The name of the destination queue manager
- The original message descriptor, that is, the one supplied by the application which put the message

If applications connected to one queue manager are putting messages on multiple queues owned by another queue manager, only one transmission queue is required to stage the delivery of these messages. Also, only one communications connection is required between the two queue managers.

Applications are shielded from the complexities of the underlying network. The application programmer does not have to be concerned with writing programs to interfaces or writing code to handle communications errors.

Asynchronous model



- Separate process for replies
- No need for communicating programs to be active at the same time
- Time independence

Figure 1-12. Asynchronous model

Notes:

In the asynchronous model, instead of waiting for a reply to its first message, Program A continues to send further requests to Program B. It is a separate process, Program X, which receives the replies when they arrive.

In this model, Program A is not dependent on Program B to be running when the requests are sent. It can continue to do work even when Program B is stopped.

The application does expect Program X to receive the replies at some time, but not necessarily at the same time that Program A or Program B. This scenario illustrates another of the major benefits of WebSphere MQ, *time independence*.

Basic MQI calls

- Send messages
  - MQPUT
  - MQPUT1
- Receive messages
  - MQGET
- Housekeeping calls
  - MQCONN, MQCONNX, and MQDISC
  - MQOPEN and MQCLOSE
- Look at or change properties
  - MQINQ and MQSET
- Perform transactions
  - MQBEGIN, MQCMIT, and MQBACK

Figure 1-13. Basic MQI calls WM203 / VM2032.0

Notes:

The most basic calls allow an application to put a message on a queue and get a message from a queue.

- **MQPUT and MQPUT1**
  Put a message on a named queue. Generally, a message is added to the end of a queue.
- **MQGET**
  Get a message from a named queue. Generally, a message is removed from the front of a queue.

The other calls are as follows:

- **MQCONN, MQCONNX, and MQDISC**
  Enables an application to connect to a queue manager and disconnect from a queue manager. An application must connect to a queue manager before it can issue any further MQI calls.

- **MQOPEN and MQCLOSE**
  Enables an application to open a queue for specified operations and close the queue when access to it is no longer required. An application must open a queue before it can access it in any way; to put messages on it, or get messages from it, for example.
- **MQINQ and MQSET**
  Inquire and set the attributes of an object. All WebSphere MQ objects, such as a queue, a process, and the queue manager object, have a set of attributes.
- **MQBEGIN, MQCMIT, and MQBACK**
  Enables an application to put and get messages as part of a unit of work.

Separate processes as units of work

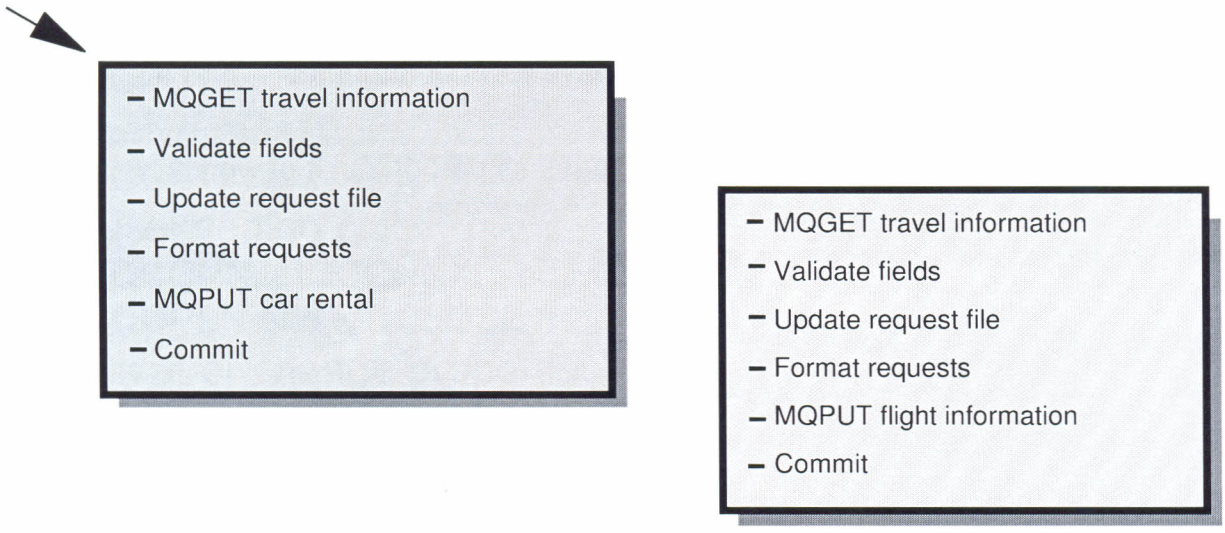


Figure 1-14. Separate processes as units of work WM203 / VM2032.0

Notes:

A complex business transaction might be implemented as a number of separate asynchronous processes where each process constitutes a unit of work. This type of design emphasizes the importance of the assured, one time delivery property of WebSphere MQ.

Multiple, asynchronous units of work

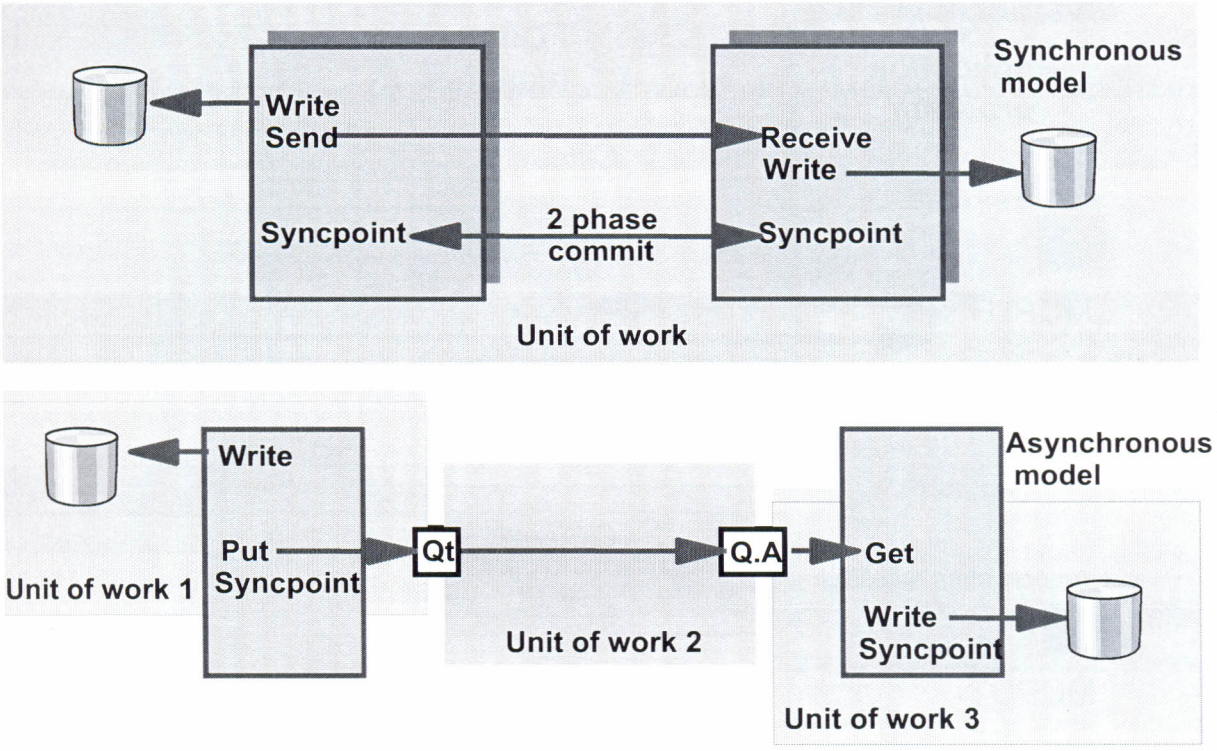


Figure 1-15. Multiple, asynchronous units of work WM203 / VM2032.0

Notes:

Implementing a business transaction as a single distributed unit of work requires a two-phase commit protocol.

WebSphere MQ encourages a different style of implementation that uses multiple units of work acting asynchronously.

Message persistence

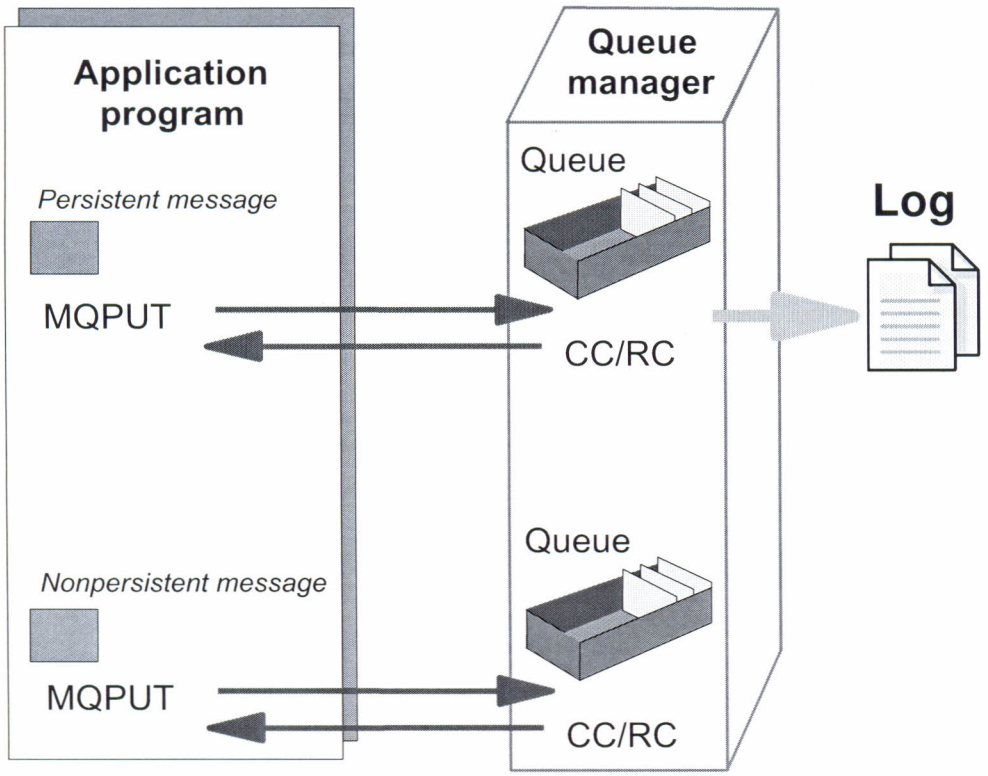


Figure 1-16. Message persistence

WM203 / VM2032.0

Notes:

A message survives a queue manager restart if that message is defined as persistent. Persistence applies even when the queue manager is stopped by an operator command or because of a system failure. The restart implies that persistent messages can be written out to a log. If a queue manager is restarted after a failure, it recovers these persistent messages as necessary from the logged data.

A message defined as nonpersistent does not survive a queue manager restart. Nonpersistence applies even if a queue manager finds an intact nonpersistent message on disk during restart. Therefore the queue manager discards it.

Both persistent and nonpersistent messages can be stored on the same queue, except for temporary dynamic queues.

Whether a message is persistent or nonpersistent is determined by the value of a field in its message descriptor.

WebSphere MQ queue manager platforms

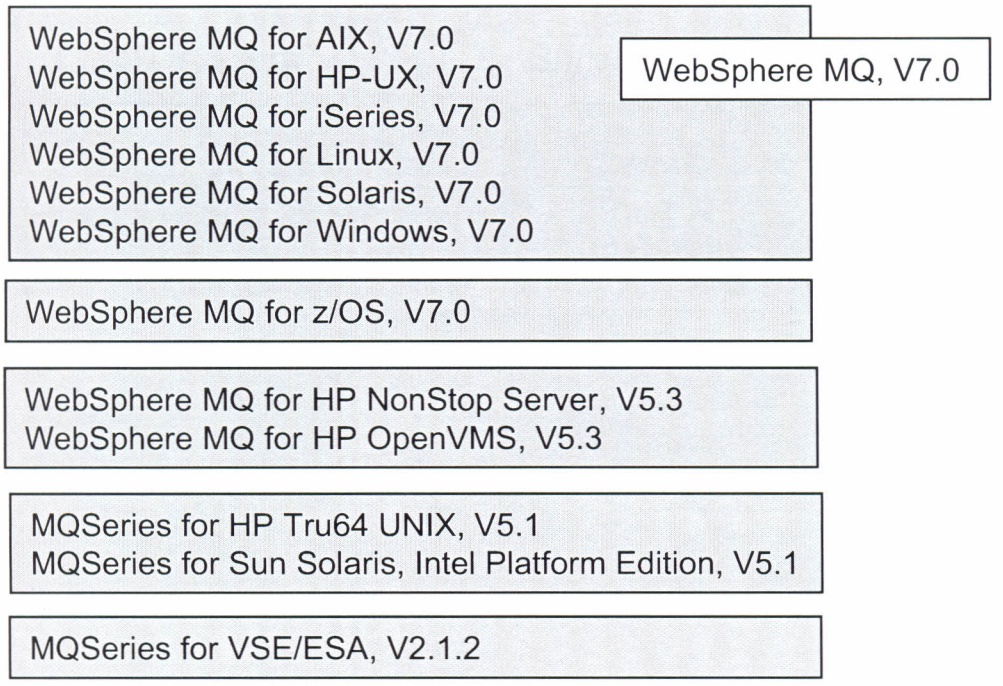


Figure 1-17. WebSphere MQ queue manager platforms WM203 / VM2032.0

Notes:

The figure includes a list of WebSphere MQ queue managers and their latest release levels supported by IBM as of the date of publication of these course notes.

Any platforms not shown here are not supported by IBM. Consult the list of platforms available on the WebSphere MQ home page for information about the source of these products.

In the remainder of these course notes, the term **UNIX systems** are used to refer to the following operating systems.

- AIX
- Compaq Tru64 UNIX
- HP-UX
- Linux
- Sun Solaris

WebSphere MQ client

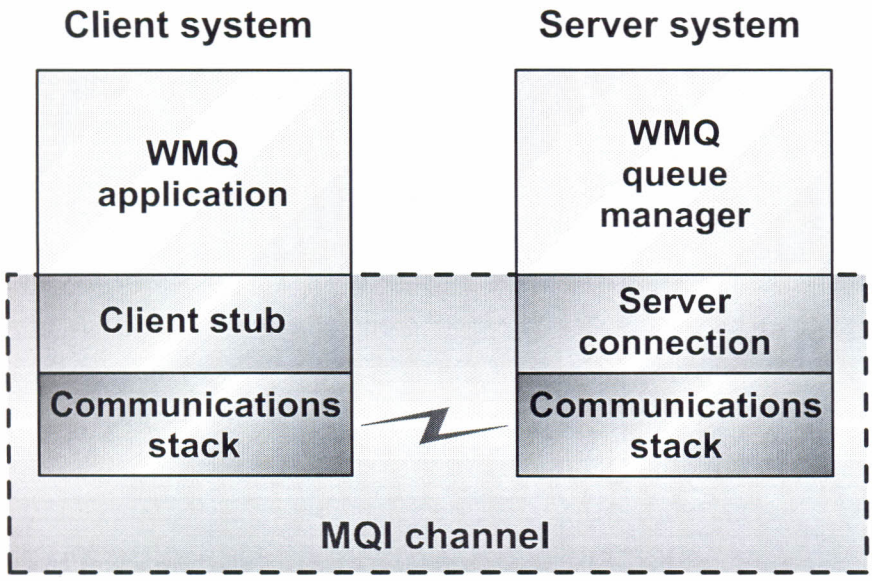


Figure 1-18. WebSphere MQ client WM203 / VM2032.0

Notes:

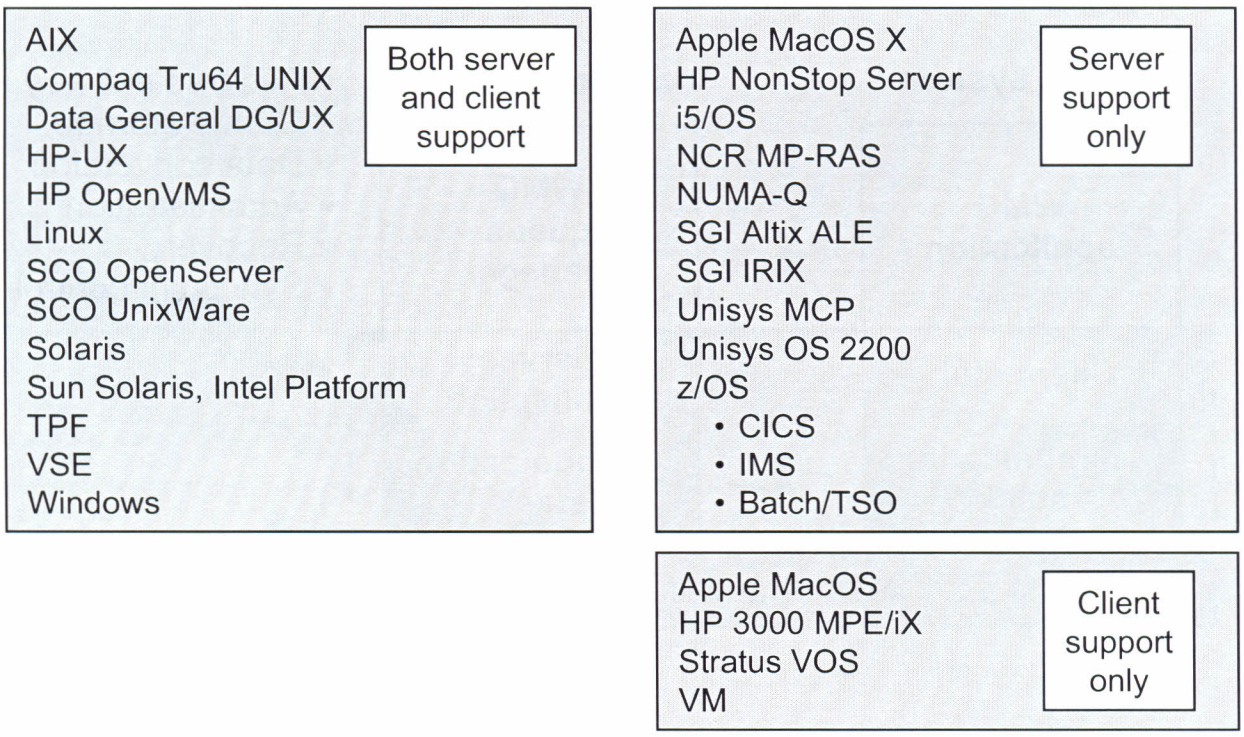
A WebSphere MQ client is a component of WebSphere MQ which allows an application running on one system to issue MQI calls to a queue manager running on another system.

The *client connection* receives the input parameters of an MQI call from the application and sends them over a communications connection to the *server connection* on the same system as the queue manager. The server connection then issues the MQI call to the queue manager on behalf of the application. After the queue manager has completed the MQI call, the server connection sends the output parameters of the callback to the client connection, which then passes them onto the application.

The combination of a client connection, a server connection, and a communications connection between them (for example, an SNA LU6.2 conversation or a TCP connection) is called an *MQI channel*.

The full range of MQI calls and options are available to a client application. The application issues an MQCONN call to connect to a queue manager, that is, to start an MQI channel.

WebSphere MQ platforms



For a more complete list of WebSphere MQ platforms, see:
http://www.ibm.com/support/docview.wss?rs=171&uid=swg27007431

Figure 1-19. WebSphere MQ platforms WM203 / VM2032.0

Notes:

A WebSphere MQ platform is a system environment in which an application issues calls to the MQI.

The platforms supported by WebSphere MQ are displayed in three groups.

- Those platforms that an application might issue MQI calls to a queue manager running on the same system, or on another system.
- Those platforms that an application might only issue MQI calls to a queue manager running on the same system.
- Those platforms that an application might only issue MQI calls to a queue manager running on another system, that is, a WebSphere MQ client application because there is no WebSphere MQ queue manager that runs on any of these platforms.

In addition to the queue managers listed earlier available from third-party vendors, there are several clients also available from other vendors. Consult the platforms listing at the WebSphere MQ home page for an up-to-date list.

Checkpoint questions

1. True or false: WebSphere MQ only supports asynchronous messaging.

2. WebSphere MQ assured delivery means that:
   a. A report of delivery can always be sent back.
   b. Unless the entire system goes down, no messages are lost.
   c. Messages can be duplicated but never lost.
   d. Messages are delivered with no loss or duplication.

3. Applications place messages on queues by use of the:
   a. WebSphere MQ program-to-program interface.
   b. WebSphere MQ message queue interface.
   c. WebSphere MQ command processor.

Figure 1-20. Checkpoint questions

Notes: