Selecting the most appropriate JMS provider for your applications.

By Jamie Roots, IBM Software Group

June 2003
Introduction

The purpose of this white paper is to help you choose the most appropriate Java™ Message Service (JMS) messaging options available from IBM, according to the requirements of your Java 2 Platform, Enterprise Edition (J2EE) applications. This paper compares the messaging technology built into IBM WebSphere® Application Server, Version 5 with IBM WebSphere MQ, IBM WebSphere Business Integration Event Broker and IBM WebSphere Business Integration Message Broker.

Application server (or transaction monitor) and messaging technologies have been used for many years. Messaging within an application server environment has several uses, including:

- **Providing asynchronous communication between applications or application components.** When a message is sent between applications, it is considered an event, and the content of the message contains data that describes the event. In point-to-point messaging, the message is sent to a specific target application. In publish and subscribe messaging, applications receive messages by subscribing to one or more topics (based on event types). Each message published to a topic is distributed to all applications that have subscribed to that topic. This type of messaging allows for asynchronous communication because unlike other forms of distributed communication, such as remote procedure calls (RPCs), the message does not control message flow from the sending to the receiving application. In J2EE applications, messages corresponding to asynchronous events can be received by the J2EE server using Message-Driven Beans (MDB), a type of Enterprise JavaBeans (EJB) technology. When a Message-Driven Bean is deployed, the destination (or queue or topic) from which it is to receive messages must be specified. The application server is notified when a message is sent to that destination (event) by the JMS provider and calls the MDB component, passing to it the message content.

- **Providing heterogeneous integration.** Disparate applications that run on various operating systems often do not have a common communications technology and must use messaging to communicate. When messaging is used for this purpose, the type of communication can be asynchronous, synchronous (RPC-style), or even batch. In all cases, the messaging transport is by nature asynchronous, as all transport protocols are built on asynchronous messaging primitives using well-established design patterns.
Providing temporary data storage. In transaction monitor environments such as IBM CICS\textsuperscript{®} applications, specific facilities are provided for storing temporary data on queues. By writing to a queue, the data can be recovered in the event of system failure. But unlike a database, the schema of the data does not have to be predefined. Many other kinds of applications also use queues as a convenient place to temporarily store data.

J2EE technology enables application developers to write application components by focusing on their functional effect (or business logic), without having to explicitly write code for transactions, concurrency and persistence. Several facilities—container-managed persistence, stateful session-bean passivation, HTTP session management—are inherently provided in J2EE technology, eliminating the need for developers to code applications explicitly for temporary data storage on queues.

WebSphere Application Server, Version 5 includes an implementation of the JMS 1.0.2 application program interface (API) as part of its support for J2EE 1.3. The JMS specification defines the standard API for Java applications to use for messaging. All products that are J2EE 1.3 technology certified, such as WebSphere Application Server, must include a JMS provider—an implementation of the JMS API—as part of the product.

Because it reduces development complexity, J2EE technology is very popular for implementing new applications, and Web services are becoming increasingly important as an integration technology. At the same time, the number of J2EE applications that need to be able to integrate with systems that don’t use J2EE or Web services continues to grow. To meet these integration needs, heterogeneous messaging that is exposed to J2EE applications through the JMS API is critical to most enterprises that use J2EE technology. WebSphere Application Server, along with other J2EE technology-based application server products, relies on WebSphere MQ or other external JMS products for heterogeneous messaging.
WebSphere MQ products

IBM WebSphere MQ is IBM’s market-leading stand-alone messaging product (formerly IBM MQSeries®), used on more than 35 hardware platforms and for point-to-point messaging from Java, C, C++ and COBOL applications. It allows Java applications to use JMS to exchange messages with Java and other applications that use the WebSphere MQ WebSphere MQ API (MQI). Three-quarters of enterprises that buy inter-application messaging systems buy WebSphere MQ, and in the largest deployment, over 250 million messages a day are transmitted.

WebSphere MQ supports both persistent and nonpersistent messaging. With persistent messaging, messages can be recovered following a system failure. Once an application has sent the message, the message can only be lost if the disk on which it is stored is damaged or lost and there are no backups or mirrored copies. With nonpersistent messages, message recovery following failure is not possible, but message throughput is considerably higher. In a typical point-to-point configuration, the sending and receiving applications each connect to their own queue managers, and the two queue managers are interconnected by means of a message channel. The sending application first delivers a message to its local queue manager. The message is then carried over the message channel to the receiving application, and last, the receiving application retrieves the message.

WebSphere Business Integration Event Broker adds support for publish and subscribe messaging. The job of a broker is to match published messages to subscriptions. A broker works in conjunction with a transport, which passes messages from the publishing application through the broker to subscribers. WebSphere Business Integration Event Broker offers two transports for use with J2EE applications to give a wide range of quality-of-service and performance options.¹ For fastest performance of nonpersistent messaging, WebSphere Business Integration Event Broker has a custom transport known as real-time transport. This real-time transport in WebSphere Business Integration Event Broker generally delivers messages as reliably as JMS products from other vendors.
## Messaging capabilities at a glance

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Point-to-point JMS</strong></td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td><strong>Publish and subscribe JMS</strong></td>
<td>X</td>
<td>IBM SupportPac (MA0C)</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td><strong>Security</strong></td>
<td>Integrated with WebSphere Application Server</td>
<td>WebSphere MQ</td>
<td>WebSphere MQ and WebSphere MQ Event Broker</td>
<td>WebSphere MQ and WebSphere MQ Integrator</td>
<td>WebSphere MQ and WebSphere Business Integration Event Broker</td>
<td>WebSphere MQ and WebSphere Business Integration Message Broker</td>
<td>WebSphere MQ and WebSphere Business Integration Message Broker with Rules and Formatter Extension</td>
</tr>
<tr>
<td><strong>Administration</strong></td>
<td>Integrated with WebSphere Application Server</td>
<td>WebSphere MQ</td>
<td>WebSphere MQ Event Broker</td>
<td>WebSphere MQ Integrator</td>
<td>WebSphere Business Integration Event Broker</td>
<td>WebSphere Business Integration Message Broker</td>
<td>WebSphere Business Integration Message Broker with Rules and Formatter Extension</td>
</tr>
<tr>
<td><strong>Run time</strong></td>
<td>Integrated with WebSphere Application Server</td>
<td>WebSphere MQ</td>
<td>WebSphere MQ Event Broker</td>
<td>WebSphere MQ Integrator</td>
<td>WebSphere Business Integration Event Broker</td>
<td>WebSphere Business Integration Message Broker</td>
<td>WebSphere Business Integration Message Broker with Rules and Formatter Extension</td>
</tr>
<tr>
<td><strong>Heterogeneous connectivity</strong></td>
<td></td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td><strong>TCP transport for lightweight messaging</strong></td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td><strong>Multicast</strong></td>
<td></td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td><strong>Multiserver topologies</strong></td>
<td>X</td>
<td>SupportPac (MA0C)</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td><strong>Multibroker topologies</strong></td>
<td></td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td><strong>Tool-driven message transformation</strong></td>
<td></td>
<td></td>
<td></td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td><strong>Tool-driven message flows</strong></td>
<td></td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
</tbody>
</table>

* Replacing IBM WebSphere MQ Event Broker, Version 2.1
** Replacing IBM WebSphere MQ Integrator Broker, Version 2.1
***Replacing IBM WebSphere MQ Integrator, Version 2.1
For more reliability, WebSphere MQ message channels can be used as the transport (WebSphere MQ is included with WebSphere Business Integration Event Broker). In this case, a stronger effort is made to avoid losing messages, even those that are in a nonpersistent state. However, for critical messages that you cannot risk losing, the WebSphere MQ transport must always be used.

**WebSphere Business Integration Message Broker**

WebSphere Business Integration Message Broker is the high-end broker product from IBM. Like WebSphere Business Integration Event Broker, it adds publish and subscribe function to WebSphere MQ capabilities. However, it also provides a rich environment for the transformation and routing of messages, including support for a wide range of XML and non-XML message formats and the integration of database access into message flows.

**WebSphere Application Server embedded messaging**

WebSphere Application Server, Version 5 includes a built-in JMS provider (or WebSphere MQ JMS provider), which is known as embedded messaging. The purpose of the WebSphere embedded messaging feature is to enable WebSphere Application Server users to exploit JMS technology without having to obtain separate messaging technology and integrate it with the WebSphere Application Server environment. Embedded messaging is built using technology from WebSphere MQ and WebSphere Business Integration Event Broker. For point-to-point messaging, the underlying transport is the same as that used by WebSphere MQ. For publish and subscribe messaging, support for both WebSphere MQ and real-time transport forms of connection between application and broker is included. The following list shows how embedded messaging differs from the stand-alone products:

- *WebSphere MQ products provide separate queue manager and broker components, whereas embedded messaging provides a single server component (the embedded messaging server option), which is accessed through a single integrated client component (the embedded messaging client option). When installing WebSphere Application Server, you can choose whether to install the messaging server option.*
• Embedded messaging is administered using the administrative console and other WebSphere Application Server administration tools. WebSphere MQ tools (such as WebSphere MQ explorer) must not be used with WebSphere embedded messaging. It is technically possible to discover the WebSphere MQ queue manager that underlies an embedded JMS provider instance and to operate on it with WebSphere MQ explorer and other WebSphere MQ tools. However, doing so is likely to put the queue manager into a state inconsistent with the application server’s view of it, which can result in undelivered messages.

• Embedded messaging is intended for use with the Web, EJB components and client containers of WebSphere Application Server, Version 5. As with other WebSphere Application Server resources, it can also be used by thin-client Java applications but does not interoperate with WebSphere MQ or WebSphere Business Integration Event Broker. Non-Java applications or Java applications that do not have access to WebSphere Application Server resources through Java Naming and Directory Interface (JNDI) cannot connect to WebSphere MQ JMS provider.

• Embedded messaging provides a specific messaging topology, as shown in Figure 1. When the embedded messaging server is installed onto a node, that node hosts a single JMS server. For two WebSphere applications to be able to communicate using JMS, they must be running in the same server cluster instance (or cell). To exchange JMS messages, the two applications must connect to the same JMS server instance. This means they must use the same JMS server node. Each node that hosts applications using JMS technology must have the embedded messaging client installed; only nodes hosting a JMS server require the embedded messaging server component.

![Figure 1. Embedded messaging topology](image-url)
• **Embedded messaging** is integrated with **WebSphere Application Server** security. User IDs are authenticated by matching the password supplied by the application or administrative configuration with the user registry that WebSphere Application Server is configured to use. The registry configuration has several options, including the underlying operating system and Lightweight Directory Access Protocol (LDAP). Access to queues (for point-to-point messaging) and topics (for publish and subscribe) is controlled by WebSphere Application Server using an XML-format JMS authorizations file.

• **The components of the embedded messaging JMS server** are integrated to form a single logical run-time server. For example, starting the JMS server causes all the run-time components associated with the underlying MQ technology to be started automatically. The JMS server is capable of both point-to-point and publish and subscribe messaging, and has no visible separation into the queue manager and broker components.

Communicating applications must use the same JMS server because those applications will be making remote connections much of the time. Previously, WebSphere MQ could only support globally coordinated XA transactions from applications running locally to the queue manager. WebSphere Application Server, Version 5 now offers enhanced WebSphere MQ technology that comprises WebSphere embedded messaging to support remote transactional connections so your applications can connect remotely to a JMS server without reduction in quality of service.

WebSphere embedded messaging is recommended for use in functional verification test, system test and small-scale production deployments. The integration of messaging technology into the application server environment means the administrative and operational cost of exploiting messaging for J2EE applications is minimized. For larger production deployments, where throughput or availability considerations require the use of a more complex messaging topology, the use of WebSphere MQ or WebSphere Business Integration Event Broker (or both) is recommended. The WebSphere MQ products are required when messaging between J2EE and non-J2EE applications is to be used.
Using WebSphere MQ and WebSphere Business Integration Event Broker with WebSphere Application Server, Version 5

By choosing to use WebSphere MQ or WebSphere Business Integration Event Broker with WebSphere Application Server, you can exploit all the features of these products with your J2EE applications. However, this means that you must separately configure the messaging system itself (using WebSphere MQ and WebSphere Business Integration Event Broker tools like runmqsc or WebSphere MQ explorer) and the connections between WebSphere Application Server and WebSphere MQ or WebSphere Business Integration Event Broker (using the WebSphere Application Server administrative console). Within WebSphere Application Server tools, WebSphere MQ and WebSphere Business Integration Event Broker are referred to collectively as WebSphere MQ JMS provider.²

To use WebSphere MQ JMS provider, you install and configure either WebSphere MQ or WebSphere Business Integration Event Broker. Next, you need to configure WebSphere Application Server to connect to the JMS provider instead of the embedded provider. WebSphere MQ products may be installed onto the same machine as WebSphere Application Server or on a different machine. It is not necessary to separately install the WebSphere MQ JMS client into the application server environment because the embedded messaging client component of the application server install provides the transactional connections to the embedded provider and WebSphere MQ JMS provider.³

If your J2EE applications use only point-to-point messaging, WebSphere MQ is sufficient. However, if they use publish and subscribe messaging, you will need WebSphere Business Integration Event Broker. WebSphere Application Server Enterprise, Version 5 includes WebSphere MQ (but not WebSphere Business Integration Event Broker), with licensing that restricts its use to WebSphere Application Server environments. For all other scenarios, the appropriate WebSphere MQ or WebSphere Business Integration Event Broker license entitlement for the messaging server in use must be obtained.
You can use additional capabilities of WebSphere MQ, WebSphere Business Integration Event Broker or WebSphere Business Integration Message Broker when you need to:

- **Connect applications running in WebSphere Application Server with applications that use a wide selection of other language environments, run-time environments and hardware platforms, or connect to a large range of packaged applications that have either a native WebSphere MQ interface or for which an adapter is available.**

- **Support high message volumes (measured as a function of both message size and number of messages). With WebSphere MQ, queue manager clustering can be used to distribute messaging workload across multiple queue managers.**

- **Decouple sending and receiving application environments, both from one another and from the underlying network that provides connectivity between them. WebSphere MQ message channels allow the sending application to continue processing when the receiving application or its hardware is unavailable—and likewise, when the sending application is unavailable—so that both applications can continue operating when the network link is down.**

- **Support a large number of independent subscribers. With WebSphere Business Integration Event Broker, multiple brokers can be interconnected to form a graph structure. This allows publications to be distributed across a large number of subscribing applications.**

- **Reuse existing WebSphere MQ or WebSphere Business Integration Event Broker infrastructure. Many organizations already have a WebSphere MQ network in place and may need to leverage their existing investments. For example, if a WebSphere MQ queue manager has been configured in a high-availability environment (such as high-availability cluster multiprocessing [HACMP] or Microsoft® Cluster Server) and has extra capacity, the applications running in WebSphere Application Server can also connect to this queue manager and benefit from its high availability. Or the existing WebSphere MQ infrastructure may have monitoring and management tools and procedures in place that are just as relevant to J2EE applications as to existing MQ applications.**
Tooling support: Messaging in WebSphere Studio Application Developer

IBM WebSphere Studio Application Developer, Version 5.0 is the build-time counterpart to WebSphere Application Server, Version 5.0 that provides the tooling necessary to develop, deploy and test WebSphere Application Server applications. Applications are tested within WebSphere Studio by running WebSphere Application Server within the WebSphere Studio integrated development environment (IDE). All deployment and configuration can be performed from within the WebSphere Studio graphical user interface (GUI).

There are many options for testing applications that use JMS messaging, including:

- **WebSphere Application Server running within the WebSphere Studio Application Developer IDE** can be configured to run its own embedded messaging JMS server.

- **Applications running in the application server inside WebSphere Studio Application Developer** can connect to an embedded messaging JMS server associated with another installation of WebSphere Application Server, usually on a different machine. This lets you employ the same technique used when one machine is installed with a database that’s dedicated to development tests, avoiding the need to install and operate the database on each developer workstation.

- **Applications running in the application server inside WebSphere Studio Application Developer** can connect to a WebSphere MQ JMS provider, as if the application server were running outside of WebSphere Studio.

- **WebSphere Studio Application Developer** can be configured to run a special JMS provider, called MQ for Java Developers. This is a subset of the first option, where a simulator for the JMS server is executed instead of the embedded messaging JMS server. This simulator implements all of the JMS APIs and has the same functional behavior, including both point-to-point and publish and subscribe capabilities, as well as the same error codes found in the embedded messaging provider. However, it does not actually implement persistence, recovery logging for transactions or security checks. MQ for Java Developers is ideal for unit testing of application components, because it allows the application server to be started and stopped very quickly—important when you have to make a lot of configuration changes early in the development cycle of an application. It should not be used for system testing or production.
Summary

IBM provides a range of options for JMS technology-based messaging in a J2EE environment that follows the start simple, grow big philosophy. For development and unit tests, MQ for Java Developers, part of IBM WebSphere Studio Application Developer, improves productivity by speeding the build-to-deploy test cycle. For system tests and small-scale production deployments, WebSphere embedded messaging, part of IBM WebSphere Application Server, provides a full JMS implementation. It is built using tried and tested technology from the WebSphere MQ family, but is fully integrated with WebSphere Application Server administration and security. For large-scale production, WebSphere MQ is a widely trusted messaging technology with many high-volume deployments around the world. WebSphere MQ also allows your J2EE applications to connect to the widest range of other platforms, environments, and applications. For publish and subscribe messaging technology, the WebSphere MQ family offers IBM WebSphere Business Integration Event Broker and WebSphere Business Integration Message Broker—to enable you to reuse your WebSphere MQ transport infrastructure or to use a custom transmission control protocol (TCP) transport optimized specifically for highly scalable, lightweight publish and subscribe messaging technology.

For more information

To learn more about how IBM WebSphere Application Server, Version 5 can work in conjunction with your messaging environment, visit:

ibm.com/software/webservers/appserv/was

To learn more about how WebSphere MQ products can enhance your messaging technology, visit:

ibm.com/software/integration/wmq
Selecting the most appropriate JMS provider for your applications.

1. This is in addition to the WebSphere MQ mobile transport, intended for mobile devices, and the SCADA transport, intended for remote telemetry devices. By using WebSphere Business Integration Event Broker for publish and subscribe messaging, J2EE applications can communicate with any WebSphere MQ mobile transport or SCADA-enabled device.

2. WebSphere Application Server also allows you to configure a generic provider. This is primarily intended for configuring WebSphere Application Server to work with JMS providers from other vendors.

3. For non-application server environments, remote transactional connections require the installation (and licensing) of a special WebSphere MQ component; without this component, only local connections or nontransactional remote connections can be made. For applications running in WebSphere Application Server, Version 5, this component is part of the embedded messaging client, and no additional WebSphere MQ client install (or licensing) is necessary.