BEA WebLogic Server®

Configuring and Managing the WebLogic Messaging Bridge

Version 9.1
Document Revised: December 16, 2005

Copyright

Restricted Rights Legend

Trademarks and Service Marks

Contents

CHAPTER 1

Introduction and Roadmap

The following sections describe the contents and organization of this guide—*Configuring and Managing the WebLogic Messaging Bridge*:

Document Scope and Audience

This document is a resource for system administrators who want to configure and manage a WebLogic Messaging Bridge as a forwarding mechanism between any two messaging products—thereby providing interoperability between separate implementations of WebLogic JMS, or between WebLogic JMS and another messaging product. It also contains information that is useful for business analysts and system architects who are evaluating WebLogic Server® or considering the use of WebLogic Server JMS for a particular application.

It is assumed that the reader is familiar with J2EE and JMS concepts. This document emphasizes the value-added features provided by WebLogic Server and key information about how to use WebLogic Server features and facilities to configure and manage a messaging bridge.

Guide to This Document

- This chapter, Chapter 1, "Introduction and Roadmap," describes the organization and scope of this guide, as well as new features and related documentation.
- Chapter 2, "Understanding the Messaging Bridge," describes basic WebLogic Messaging Bridge resources, such as resource adapters and destinations.
- Chapter 3, "Designing a Messaging Bridge," explains design options and other prerequisite considerations for configuring a WebLogic Messaging Bridge.
- Chapter 4, "Interoperating with Different WebLogic Server Releases or Foreign Providers," explains the interoperability guidelines that apply when using the messaging bridge to access JMS destinations on different releases of WebLogic Server and in other WebLogic Server domains.
- Chapter 5, "Tuning WebLogic Message Bridge," provides information on how to tune a messaging bridge to achieve improved performance.

Related Documentation

For information on topics related to configuring and managing a messaging bridge, see the following documents:

- *Configuring and Managing WebLogic JMS* is a guide to configuring and managing WebLogic JMS resources.
- *Programming WebLogic JMS* is a guide to developing WebLogic JMS applications.
- *Developing Applications with WebLogic Server* is a guide to developing WebLogic Server applications.
- *Deploying Applications to WebLogic Server* is the primary source of information about deploying WebLogic Server applications.
- *Programming WebLogic Resource Adaptors* contains information on WebLogic resource adapters and the BEA WebLogic Server implementation of the J2EE Connector Architecture.

Samples and Tutorials for the JMS Developer

In addition to this document, BEA Systems provides a variety of code samples and tutorials for JMS developers. The examples and tutorials illustrate WebLogic Server JMS in action, and provide practical instructions on how to perform key JMS development tasks.

BEA recommends that you run some or all of the JMS examples before developing your own EJBs.

Avitek Medical Records Application (MedRec) and Tutorials

MedRec is an end-to-end sample J2EE application shipped with WebLogic Server that simulates an independent, centralized medical record management system. The MedRec application provides a framework for patients, doctors, and administrators to manage patient data using a variety of different clients.

MedRec demonstrates WebLogic Server and J2EE features, and highlights BEA-recommended best practices. MedRec is included in the WebLogic Server distribution, and is accessed from the Start menu on Windows machines. For Linux and other platforms, start MedRec from the `WL_HOME\samples\domains\medrec` directory, where `WL_HOME` is the top-level installation directory for WebLogic Platform.

MedRec includes a service tier comprised primarily of Enterprise Java Beans (EJBs) that work together to process requests from web applications, web services, and workflow applications, and future client applications. The application includes message-driven, stateless session, stateful session, and entity EJBs.

JMS Examples in the WebLogic Server Distribution

WebLogic Server 9.1 optionally installs API code examples in `WL_HOME\samples\server\examples\src\examples`, where `WL_HOME` is the top-level directory of your WebLogic Server installation. Start the examples server and obtain information about the samples and how to run them from the WebLogic Server 9.1 Start menu.

Additional JMS Examples Available for Download

Additional API examples for download at http://dev2dev.bea.com/code/index.jsp. These examples are distributed as.zip files that you unzip into an existing WebLogic Server samples directory structure. You build and run the downloadable examples in the same manner as you would an installed WebLogic Server example.

New and Changed JMS Features in This Release

Many new Messaging Bridge features were introduced in WebLogic Server 9.0. For information on these features see the What's New in WebLogic Server 9.0 section of the *WebLogic Server Release Notes*.

CHAPTER 2

Understanding the Messaging Bridge

The following sections describe WebLogic Messaging Bridge concepts and functionality:

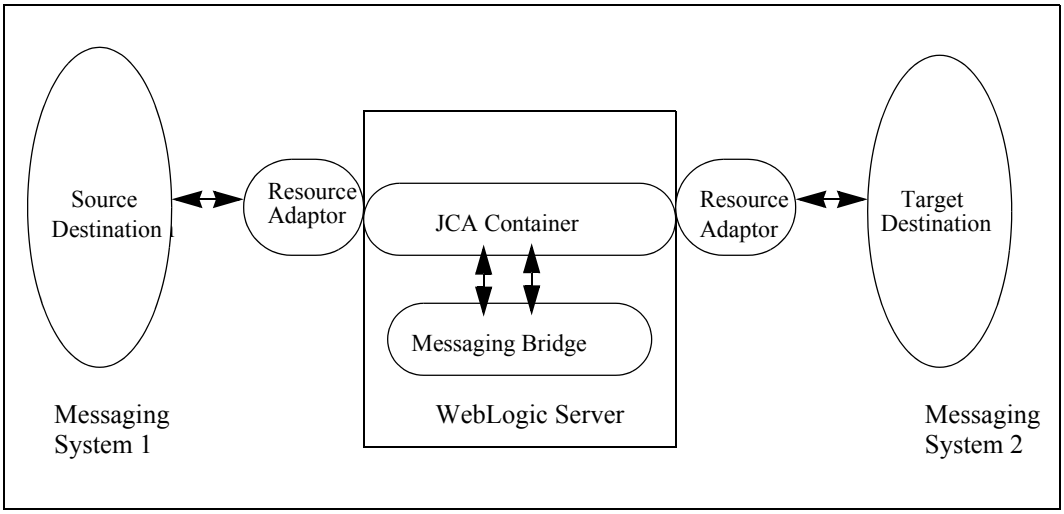
What Is a Messaging Bridge?

The WebLogic Messaging Bridge is a forwarding mechanism that provides interoperability between WebLogic JMS implementations, and between JMS and other messaging products. Use the Messaging Bridge to integrate your messaging applications between:

- Any two implementations of WebLogic JMS, including those from separate releases of WebLogic Server.
- WebLogic JMS implementations that reside in separate WebLogic domains.
- WebLogic JMS and a third-party JMS product (for example, MQSeries).

A messaging bridge instance forwards messages between a pair of bridge source and target destinations. These destinations are mapped to a pair of bridge source and target destinations. The messaging bridge reads messages from the source bridge destination and forwards those messages to the target bridge destination. For WebLogic JMS and third-party JMS products, a messaging bridge communicates with source and target destinations using the J2EE Connector Architecture (JCA) resource adapters provided with WebLogic Server. See Figure 2-1.

Figure 2-1 WebLogic Messaging Bridge



You designate source and target bridge destinations as either queues, topics, or distributed destinations. Optionally, specify a quality of service (QOS), as well as message filters, transaction semantics, and connection retry policies. Once you configure a messaging bridge you can perform management tasks easily from the Administration Console, including suspending bridge traffic temporarily whenever necessary and monitoring the status of all your configured bridges.

Messaging Bridge Components

The following sections describe resources you need to use a messaging bridge:

- "Resource Adapters" on page 2-2
- "Source and Target Bridge Destinations" on page 2-6
- "Messaging Bridge Instance" on page 2-7

Resource Adapters

A messaging bridge uses JCA resource adapters to communicate with the configured source and target JMS destinations. You need to associate both the source and target JMS destinations with a *supported* resource adapter in order for the bridge to communicate with them. The JNDI name

for the adapter is configured as part of the resource adapter's deployment descriptor. See Table 2-1, "Messaging Bridge Resource Adapters and JNDI Names," on page 2-5.

Resource adapters for different types of JMS destinations are provided in exploded format or in a `.rar` file. The exploded format gives you an easy way to modify resource adapter deployment descriptor parameters, such as the `max-capacity` of the connection factory that specifies the maximum number of connections available for bridge instances.

Note: Changing a deployment descriptor for a resource adapter using the exploded format does not update the descriptor packaged in the `.rar` file. See "Setting the Number of Connection Factories" on page 3-4.

The supported resource adapters are located in the *WL_HOME*\server\lib directory and are described in the following table.

Table 2-1 Messaging Bridge Resource Adapters and JNDI Names

Adapter	JNDI Name	Description
`jms-xa-adp`	`eis.jms.WLSConnectionFactoryJNDIXA`	Provides transaction semantics via the `XAResource`. Used when the required QOS is *Exactly-once*. This envelops a received message and sends it within a user transaction (XA/JTA). The following requirements apply to use of this resource adapter: • Any WebLogic Server implementation being bridged must be release 6.1 or later. • The source and target JMS connection factories must be configured to use the `XAConnectionFactory`. **Note:** Before deploying this resource adapter, refer to the "Access Destinations In a Release 6.1 or Later Domain" on page 4-3 for specific transactional configuration requirements and guidelines.

Table 2-1 Messaging Bridge Resource Adapters and JNDI Names

Adapter	JNDI Name	Description
`jms-notran-adp`	`eis.jms.WLSConnectionFactoryJNDINoTX`	Provides no transaction semantics. Used when the required QOS is *Atmost-once or Duplicate-okay*. If the requested QOS is *Atmost-once*, the resource adapter uses `AUTO_ACKNOWLEDGE` mode. If the requested QOS is *Duplicate-okay*, `CLIENT_ACKNOWLEDGE` is used. **Note:** For more information about the acknowledge modes used in non-transacted sessions, see "Understanding WebLogic JMS" in *Programming WebLogic JMS*.
`jms-notran-adp51`	`eis.jms.WLS51ConnectionFactoryJNDINoTX`	Provides interoperability with release 5.1 when either the source or target destination is a 5.1 server instance. This resource adapter provides no transaction semantics; therefore, it only supports a QOS of *Atmost-once* or *Duplicate-okay*. If the requested QOS is *Atmost-once*, the resource adapter uses the `AUTO_ACKNOWLEDGE` mode. If the requested QOS is *Duplicate-okay*, `CLIENT_ACKNOWLEDGE` is used. Deprecated, see "WebLogic Server 5.1 End-of-Life Announcement" in *Supported Configurations*.

Source and Target Bridge Destinations

A messaging bridge connects two actual destinations that are mapped to bridge destinations: a source destination *from which* messages are received, and a target destination *to which* messages are sent. For JMS messaging products, whether it is a WebLogic JMS implementation or a third-party JMS provider, you need to configure a JMS bridge destination instance for each actual source and target JMS destination being mapped to a messaging bridge. A JMS bridge destination

instance defines a unique name for a bridge's source and target destinations within a WebLogic domain; the name of the adapter used to communicate with the specified destination; property information to pass to the adapter (such as Connection URL and Connection Factory JNDI Name), and, optionally, a user name and password. See "Create JMS bridge destinations" in *Administration Console Online Help*.See:

- "Interoperating with Different WebLogic Server Releases" on page 4-1 or "Interoperating with Foreign Providers" on page 4-5 sections for specific configuration requirements and guidelines.
- When configuring third-party JMS provider bridge destination, use the Foreign JMS Server feature to configure multiple source or target destinations quickly. See Simplified Access to Remote or Foreign JMS Providers in *Programming WebLogic JMS*.

Messaging Bridge Instance

A messaging bridge instance communicates with the configured source and target bridge destinations. For each mapping of a source destination to a target destination, whether it is another WebLogic JMS implementation or a third-party JMS provider, you must configure a messaging bridge instance. Each messaging bridge instance defines the source and target destination for the mapping, a message filtering selector, a QOS, transaction semantics, and various reconnection parameters. See "Create Messaging Bridge Instances" in *Administration Console Online Help*.

Configuring and Managing a Messaging Bridge

The following sections provide information on how to use the Administration Console to configure and manage a messaging bridge:

- "Create a Messaging Bridge Instance" on page 2-7
- "Manage a Messaging Bridge Instance" on page 2-8

Create a Messaging Bridge Instance

Creating a messaging bridge consists of the following tasks:

1. Create source and target bridge destinations.
2. Deploy a resource adapter.
3. Create a messaging bridge instance.

4. Target the messaging bridge.

The Administration Console assists you in creating a messaging bridge by deploying an appropriate resource adapter and setting the values of some attributes. You may need to change messaging bridge settings to better suit your environment. See "Create Messaging Bridge Instances" in *Administration Console Online Help*.

Manage a Messaging Bridge Instance

Typical tasks required to manage a messaging bridge using the Administration Console include:

- Monitoring the status of all configured messaging bridges in your domain. See "Monitor messaging bridges" in *Administration Console Online Help*.
- Suspending and restarting an active messaging bridge. See "Suspend and restart messaging bridges" in *Administration Console Online Help*.
- Configuring the default execute thread pool size for your messaging bridges. See "Configure messaging bridge execute thread pool size" in *Administration Console Online Help*.
- Deploying a resource adapter. See "Deploy resource adapters" in *Administration Console Online Help*.
- Creating a trusted security relationship. See "Configure trusted relationships" in *Administration Console Online Help*.

CHAPTER 3

Designing a Messaging Bridge

Use the following information to help you design and configure a WebLogic Messaging Bridge:

When to use a Messaging Bridge

The following sections provide information on when to use a messaging bridge:

Store and Forward Messaging

A messaging bridge provides high availability for remote destinations. Store and forward messaging enables a local client to produce to a local destination and have those messages automatically forwarded to the remote destination when it is available. This allows a local client to continue to produce messages when a remote destination is not available. See "Messaging Persistence" on page 3-3.

Use the WebLogic Messaging Bridge to provide an administrative solution to store and forward messages between:

- Any two implementations of WebLogic JMS, including those from separate releases of WebLogic Server.
- WebLogic JMS implementations that reside in separate WebLogic domains.
- WebLogic JMS with a third-party JMS product (for example, MQSeries).

Replicating a Topic

A messaging bridge can be used to replicate a topic, similar to using the distributed topics feature available in WebLogic Server releases 7.0 and higher, consequently improving scalability and high availability in some scenarios. Topic replication is accomplished by configuring the bridge to subscribe to one topic and forward the topic's messages to another topic, in essence creating two topics with the same message stream. See "Create messaging bridge instances" in *Administration Console Online Help.*

When to Avoid using a Messaging Bridge

The following sections provide information on when to avoid using messaging bridge:

- Receiving messages from a remote destination—Use a message driven EJB or implement a client consumer directly.
- Sending messages to a local destination—Send directly to the local destination.
- Environment with low tolerance for message latency. Messaging Bridges increase latency and may lower throughput. Messaging bridges increase latency for messages as they introduce an extra destination in the message path and may lower throughput because they forward messages using a single thread.
- Forward messages between WebLogic 9.0 domains—Use WebLogic Store-and-Forward.

The following table summarizes information on when to use WebLogic Messaging Bridge or other forwarding technologies:

Table 3-1 Comparing Message Forwarding Technologies

Feature	Messaging Bridge	Message Driven Beans	WebLogic Store-and-Forward
Implementation mechanism	Administrative	Programmatic	Administrative
Support for foreign and legacy providers	Yes	Yes	No, use to forward messages between WebLogic 9.0 domains.

Selecting a Quality-of-Service (QOS) Level

The WebLogic Messaging Bridge supports three different QOS levels:

- Exactly-once—The highest QOS guarantees that a message is sent to the remote endpoint once and only once. With Exactly-once, messages survive server crashes and network down time, while guaranteeing one occurrence of each message at the endpoint.
- At-least-once—Guarantees that a message is sent to the remote endpoint, but with the possibility of duplicates. With At-least-once, multiple copies of a message might show up on the remote endpoint because of network failures or server crashes that occur when the message is in transit.
- At-most-once—The lowest QOS guarantees that each message is sent to the remote endpoint only once, if at all. It does not guarantee that a message is sent to the endpoint. With At-most-once, messages may get lost because of network failures or server crashes. No duplicate messages will reach the endpoint.

In some instances, the target destination may not be able to provide the quality of service configured for the bridge. In these cases, configure the bridge instance to allow the quality of service to be degraded by setting the `QOSDegradationAllowed` flag. See "Create messaging bridge instances" in *Administration Console Online Help*.

Messaging Persistence

Store-and-forward messaging enables a local JMS client to produce messages to a local destination and have those messages automatically forwarded to a remote destination when it is

available. The bridge will forward these messages to the target destination when it is restarted. A messaging bridge will store-and-forward messages to a target destination under the following conditions:

- The source destination is a queue.
- The source destination is a topic *and* the `Durability Enabled` attribute is set. This creates a durable subscription. For more information configuring durable topic subscribers, see "Setting Up Durable Subscriptions" in *Programming WebLogic JMS*.

Message Ordering

If an application message is in a transaction, saving the message in the persistent store must be part of the user transaction to preserve exactly-once semantics. In particular, the message is removed from the persistent store as part of the transaction rollback if the application decides to rollback the transaction. However, forwarding is *not* part of the application transaction. The sending agent does not forward a transactional message until the transaction commits. Within a transaction, message ordering is preserved based on when the messages are sent.

To ensure message ordering, configure a message unit-of-order. See "Using Message Unit-of-Order" in *Programming WebLogic JMS*.

Setting the Number of Connection Factories

You may need to modify the capacity of the connection factory associated with each resource adaptor by adjusting the `initial-capacity` and `max-capacity` attributes the `weblogic-ra.xml` descriptor file. In general, the value of the `max-capacity` attribute should be at least two times the number of bridge instances. For example:

> The default configuration sets the value of the `max-capacity` attribute to 20. This setting is adequate for environments that have up to ten message bridge instances targeted. If you increase the number of bridge instances to 15, increase the `max-capacity` attribute to 30.

Use the following steps to modify the `weblogic-ra.xml` descriptor file:

1. Using the editor of your choice, update the attribute with the desired value. See Listing 3-1.
2. Deploy the updated adapter.
3. Stop and restart any bridge instance that requires the new values.

Listing 3-1 Example weblogic-ra.xml Descriptor File

```
<weblogic-connection-factory-dd>


     <connection-factory-name>WLSJMSConnectionFactoryLocal</connection-fac
tory-name>
     <jndi-name>eis/jms/WLSConnectionFactoryJNDILocal</jndi-name>
          <pool-params>
               <initial-capacity>0</initial-capacity>
               <max-capacity>20</max-capacity>
          </pool-params>


</weblogic-connection-factory-dd>
```

Preserving Message Properties

Set `PreserveMsgProperty` to preserve message properties in a message header when a message is forwarded by a bridge instance. In previous releases, message properties are inherited from the `Default Delivery Mode` attribute on the connection factory used when a message is forwarded to its target destination. If the `Default Delivery Mode` is persistent, a non-persistent message is forwarded as a persistent message resulting in a significant performance loss.

When `PreserveMsgProperty` is enabled, an incoming non-persistent message is forwarded by the bridge to the target destination as a non-persistent message and an incoming persistent message is forwarded to the target destination as a persistent message. See "Configure messaging bridge instances" in *Administration Console Online Help*.

The behavior of a messaging bridge instance is determined according to the following guidelines:

- The `PreserveMsgProperty` is not enabled. This setting provides the same forwarding behavior as previous releases.
- The default value of `PreserveMsgProperty` when configuring a messaging bridge instance is not enabled.

- The `PreserveMsgProperty` is enabled. Message properties are preserved as described the following table:

Table 3-2 Message Properties Preserved at Target Destination by WebLogic Server Release

Property	WebLogic Server 9.0 and Higher	WebLogic Server 8.1, 7.0, and 6.x	WebLogic Server 5.1	Foreign JMS Servers
Message ID	Yes	No	No	No
Timestamp	Yes	No	No	No
User ID	Yes	No	No	No
Delivery Mode	Yes	Yes	No	Yes
Priority	Yes	Yes	No	Yes
Expiration Time	Yes	Yes	No	Yes
Redelivery Limit	Yes	No	No	No
Unit-of-Order name	Yes	No	No	No

Using the JMSXUserID Property with a Messaging Bridge

The messaging bridge does not disclose a message's `JMSXUserID` across messaging bridge boundaries. A JMSXUserID is a system generated property that identifies the user sending the message, see the JMS Specification.

Using Distributed Destinations as Source and Target Destinations

A messaging bridge can send to and receive from distributed destinations. BEA Systems recommends the following configurations:

- If the source is a distributed destination, the bridge is pinned to one of the members when it connects to the destination. It stays connected to that member until an event occurs that breaks the connection. On reconnection, the bridge uses the next available member. Once a bridge is connected, it does not receive messages from other members of the distributed destination. It is a best practice to configure one bridge for each member of a distributed destination using the member's JNDI Name.

- If the target is a distributed destination, the best practice is to send to the distributed destination using the distributed destination's JNDI Name and disable server affinity. This allows the distributed destination to load balance incoming messages. See "Load Balancing for JMS" in *Using WebLogic Server Clusters*.

CHAPTER 4

Interoperating with Different WebLogic Server Releases or Foreign Providers

The following sections provide interoperability guidelines for using the WebLogic Messaging Bridge to access JMS destinations on different releases of WebLogic Server or when accessing foreign providers:

Interoperating with Different WebLogic Server Releases

The following interoperability guidelines apply when using the messaging bridge to access JMS destinations on different releases of WebLogic Server and in other WebLogic Server domains.

Note: When the messaging bridge is used to communicate between two domains running different releases of WebLogic Server, BEA recommends that the messaging bridge be configured to run on the domain using the latest release of WebLogic Server.

Naming Guidelines for WebLogic Servers and Domains

Unique naming rules apply to all WebLogic Server deployments if more than one domain is involved. Therefore, make sure that:

- WebLogic Server instances and domain names are unique.
- WebLogic JMS server names are unique name across domains.
- If a JMS file store is being used for persistent messages, the JMS file store name must be unique across domains.

Configuring Interoperability for WebLogic Domains

Unless the Exactly-once QOS (quality of service) is required for handling two-phase transactions sent across the messaging bridge, there are no special security configuration requirements for the bridge to interoperate between two release 6.1 or later domains.

However, you *must* follow these steps when a bridge running on release 7.0 domain must handle transactional messages (using the Exactly-once QOS) between two release 6.1 or later domains

1. For each server participating in the transaction, set the `Security Interoperability Mode` flag according to Setting Security Interoperability Mode in *Programming WebLogic JTA* before rebooting.

   Note: When `Security Interoperability Mode` is set to performance, you are not required to set domain trust between the domains.

2. Configure domain trust for the all participating bridge domains.

   a. In all participating WebLogic Server 6.x domains, change the password for the `system` user to the same value in all participating domains on the Security→Users tab in the Administration Console. See Changing the System Password.

   b. Establish domain trust by setting a security credential for all domains to the same value in all participating domains. If you have participating 6.x domains, set the security credential for all domains to the same value as the `system` password in all participating WebLogic Server 6.x domains.

   – For 7.x domains, see Enabling Trust Between WebLogic Domains in *Managing WebLogic Security.*

   – For 8.x domains, see Enabling Trust Between WebLogic Domains in *Managing WebLogic Security.*

  - For 9.x domains, see Enable trust between domains in *Administration Console Online Help*.

Access Destinations In a Release 6.1 or Later Domain

Use these guidelines when configuring a messaging bridge on a release 9.0 domain to provide "Exactly-once" transactional message communication between two release 6.1 or later domains.

Note: The *Exactly-once* quality of service for two-phase transactions is only supported for release 6.1 or later.

- If a JMS file store is being used for persistent messages, the JMS file store name must be unique across WebLogic domains, as described in "Naming Guidelines for WebLogic Servers and Domains" on page 4-2.
- Make sure that security interoperability between the domains is correctly configured, as described in "Configuring Interoperability for WebLogic Domains" on page 4-2.
- Make sure that the XA connection factory is enabled for the domains by selecting the XAConnection Factory Enabled check box. See "Configure connection factory transaction parameters" in *Administration Console Online Help*.
- Deploy the transaction resource adapter, `jms-xa-adp.rar`, on the 9.0 domain where the messaging bridge is running, as described in "Deploy resource adaptors" in *Administration Console Online Help*.
- When configuring the JMS bridge destinations, as described in "Create JMS bridge destinations" in *Administration Console Online Help*, do the following for both the source and target destinations:
  - In the Adapter JNDI Name field, identify the transaction adapter's JNDI name, `eis.jms.WLSConnectionFactoryJNDIXA`.
  - Do not enter anything in the Adapter Classpath field.
- Select a Quality Of Service of *Exactly-once*, as described in "Configure messaging bridge instances" in *Administration Console Online Help*.

Access Destinations In a Release 5.1 Domain

Note: The `jms51-interop.jar` file and `jms-notran-adp51.rar` file are deprecated, see "WebLogic Server 5.1 End-of-Life Announcement" in *Supported Configurations*.

When configuring a messaging bridge involves interoperability between WebLogic Server 9.0 and release 5.1, you must configure the following on the WebLogic Server 9.0 implementation that the bridge is running on:

- The *Exactly-once* QOS for transactions is not supported for WebLogic Server 5.1. For more information on the bridge QOS options, see the Attribute table in "Messaging Bridge: Configuration: General".

- The `jms51-interop.jar` file in the *`WL_HOME`*`\server\lib` directory must be in the `CLASSPATH` of the WebLogic Server 9.0 implementation.

- The release 5.1 resource adapter (`jms-notran-adp51.rar`) and the non-transaction adapter (`jms-notran-adp.rar`) must be deployed on the 9.0 bridge domain, as described in "Deploy resource adapters" in *Administration Console Online Help*.

- When configuring the JMS source and target destinations, as described in "Create JMS bridge destinations" in *Administration Console Online Help*, do the following:

  In the Adapter JNDI Name field:

  - For the 9.0 destination, specify the non-transaction adapter's JNDI name as `eis.jms.WLSConnectionFactoryJNDINoTX`.
  - For the 5.1 destination, specify the 5.1 adapter's JNDI name as `eis.jms.WLS51ConnectionFactoryJNDINoTX`.

  In the Adapter Classpath field:

  - For the 9.0 destination, leave the field blank.
  - For the 5.1 destination, indicate the location of the classes for the WebLogic Server 5.1 release, as well as the location of the `jms51-interop.jar` file for the 9.0 release.

    For example, if you have WebLogic Server 5.1 GA installed in a directory named *`WL51_HOME`* and your WebLogic Server 9.0 release is installed in *`WL81_HOME`*, then set the Adapter Classpath as follows for the 5.1 destination:

    `WL51_HOME``\classes;`*`WL51_HOME`*`\lib\`*`weblogicaux.jar`*`;`
    `WL81_HOME``\server\lib\jms51-interop.jar`

    Note: If your implementation is using a 5.1 Service Pack, the corresponding *`sp.jar`* files must also be added to the Adapter Classpath field.

- Select a Quality Of Service of *Atmost-once* or *Duplicate-okay*, as described in "Configure messaging bridge instances" in *Administration Console Online Help*.

Interoperating with Foreign Providers

When configuring a messaging bridge involves interoperability with a third-party messaging provider, you must configure the following:

- Before starting WebLogic Server:
  - Supply the provider's `CLASSPATH` in the WebLogic Server `CLASSPATH`.
  - Include the `PATH` of any native code required by the provider's client-side libraries in the WebLogic Server system `PATH`. (This variable may vary depending on your operating system.)
- In the `JMSBridgeDestination` instance for the third-party messaging product being bridged, provide *vendor-specific* information in the following attributes:
  - Connection URL
  - Initial Context Factory
  - Connection Factory JNDI Name
  - Destination JNDI Name

Note: The messaging bridge cannot provide the "Exactly-once" quality of service when the source and target bridge destinations are located on the same resource manager (that is, when the bridge is forwarding a global transaction that is using the XA resource of the resource manager). For example, when using MQ Series, it is not possible to use the same Queue Manager for the source and target bridge destinations.

For more information on configuring the remaining attributes for a JMS Bridge Destination, see "Create JMS Bridge destinations" in *Administration Console Online Help*.

CHAPTER 5

Tuning WebLogic Message Bridge

The main objective when tuning a messaging bridge is to improve overall messaging performance. Raw speed, though important, is only one of several performance-related factors. Other performance factors include reliability, scalability, manageability, monitoring, user transactions, message-driven bean support, and integration with an application server.

The following sections provide information on various methods to improve message bridge performance:

Changing the Batch Size

When the `Asynchronous Mode Enabled` attribute is set to false and the quality of service is `Exactly-once`, the `Batch Size` attribute can be used to reduce the number of transaction

commits by increasing the number of messages per transaction (batch). The best batch size for a bridge instance depends on the combination of JMS providers used, the hardware, operating system, and other factors in the application environment. See "Configure transaction properties" in *Administration Console Online Help.*

Changing the Batch Interval

When the `Asynchronous Mode Enabled` attribute is set to false and the quality of service is `Exactly-once`, the `BatchInterval` attribute is used to adjust the amount of time the bridge waits for each batch to fill before forwarding batched messages. The best batch interval for a bridge instance depends on the combination of JMS providers used, the hardware, operating system, and other factors in the application environment. For example, if the queue is not very busy, the bridge may frequently stop forwarding in order to wait batches to fill, indicating the need to reduce the value of the `BatchInterval` attribute. See "Configure transaction properties" in *Administration Console Online Help.*

Changing the Quality of Service

An `Exactly-once` quality of service may perform significantly better or worse than `At-most-once` and `Duplicate-okay`.

When the `Exactly-once` quality of service is used, the bridge must undergo a two-phase commit with both JMS servers in order to ensure the transaction semantics and this operation can be very expensive. However, unlike the other qualities of service, the bridge can batch multiple operations together using `Exactly-once` service.

You may need to experiment with this parameter to get the best possible performance. For example, if the queue is not very busy or if non-persistent messages are used, `Exactly-once` batching may be of little benefit. See "Configure messaging bridge instances" in *Administration Console Online Help.*

Using Multiple Bridge Instances

If message ordering is not required, consider deploying multiple bridges.

Multiple instances of the bridge may be deployed using the same destinations. When this is done, each instance of the bridge runs in parallel and message throughput may improve. If multiple bridge instances are used, messages will not be forwarded in the same order they had in the source destination. See "Create messaging bridge instances" in *Administration Console Online Help.*

Consider the following factors when deciding whether to use multiple bridges:

- Some JMS products do not seem to benefit much from using multiple bridges
- WebLogic JMS messaging performance typically improves significantly, especially when handling persistent messages.
- If the CPU or disk storage is already saturated, increasing the number of bridge instances may decrease throughput.

Changing the Thread Pool Size

A general bridge configuration rule is to provide a thread for each bridge instance targeted to a server instance. Use one of the following options to ensure that an adequate number of threads is available for your environment:

- Use the common thread pool—A server instance changes its thread pool size automatically to maximize throughput, including compensating for the number of bridge instances configured. See Understanding How WebLogic Server Uses Thread Pools in *Designing and Configuring WebLogic Server Environments.*
- Configure a work manager for the `weblogic.jms.MessagingBridge` class. See Understanding Work Managers in *Designing and Configuring WebLogic Server Environments.*
- Use the Administration console to set the `Thread Pool Size` property in the Messaging Bridge Configuration section on the **Configuration: Services** page for a server instance. To avoid competing with the default execute thread pool in the server, messaging bridges share a separate thread pool. This thread pool is used only in synchronous mode (`Asynchronous Mode Enabled` is not set). In asynchronous mode the bridge runs in a thread created by the JMS provider for the source destination. Deprecated in WebLogic Server 9.0.

Avoiding Durable Subscriptions

If the bridge is listening on a topic and it is acceptable that messages are lost when the bridge is not forwarding messages, disable the `Durability Enabled` flag to ensure undeliverable messages do not accumulate in the source server's store. Disabling the flag also makes the messages non-persistent. See "Configure messaging bridge instances" in *Administration Console Online Help.*

Co-locating Bridges with Their Source or Target Destination

If a messaging bridge source or target is a WebLogic destination, deploy the bridge to the same WebLogic server as the destination. Targeting a messaging bridge with one of its destinations eliminates associated network and serialization overhead. Such overhead can be significant in high-throughput applications, particularly if the messages are non-persistent.

Changing the Asynchronous Mode Enabled Attribute

The `Asynchronous Mode Enabled` attribute determines whether the messaging bridge receives messages asynchronously using the JMS `MessageListener interface`, or whether the bridge receives messages using the synchronous JMS APIs. In most situations, the `Asynchronous Enabled` attributes value is dependent on the QOS required for the application environment as shown in Table 5-1:

Table 5-1 Asynchronous Mode Enabled Values for QOS Level

QOS	Asynchronous Mode Enabled Attribute value
`Exactly-once`[1]	false
`At-least-once`	true
`At-most-once`	true

1. If the source destination is a non-WebLogic JMS provider and the QOS is Exactly-once, then the Asynchronous Mode Enabled attribute is disabled and the messages are processed in asynchronous mode.

See "Configure messaging bridge instances" in *Administration Console Online Help*.

Synchronous Mode

A quality of service of `Exactly-once` has a significant effect on bridge performance. The bridge starts a new transaction for each message and performs a two-phase commit across both JMS servers involved in the transaction. Since the two-phase commit is usually the most expensive part of the bridge transaction, as the number of messages being processed increases, the bridge performance tends to decrease.

Note: In WebLogic Server 9.1, your synchronous consumers can also use the same efficient behavior as asynchronous consumers by enabling the Prefetch Mode for Synchronous

Consumers option on JMS connection factories, as described in Receiving Messages Synchronously in *Programming WebLogic JMS*.

Asynchronous Mode

In situations where a quality of service of `Exactly-once` is not required, the source destination can pipeline or process multiple messages in each transaction. If an application uses asynchronous consumers (such as MDBs), consider increasing the WebLogic JMS Connection Factory's configured `MessagesMaximum` value.

Note: Some non-WebLogic JMS implementations do not pipeline messages, or optimize the asynchronous listeners, in which case the `Asynchronous Mode Enabled` parameter may have little effect.

If the asynchronous pipeline size is already set to 1, this may indicate that the application has enabled ordered redelivery, which in turn means that the pipeline must not be increased. See "Ordered Redelivery of Messages" in *Programming WebLogic JMS*.

WebLogic JMS pipelines messages are delivered to asynchronous consumers, otherwise known as message listeners. This action aids performance because messages are aggregated when they are internally pushed from the server to the client. The messages backlog (the size of the pipeline) between the JMS server and the client is tunable by configuring the `MessagesMaximum` setting on the connection factory. See "Asynchronous Message Pipeline" in *Programming WebLogic JMS*.

In some circumstances, tuning the `MessagesMaximum` parameter may improve performance dramatically, such as when the JMS application defers acknowledges or commits. In this case, BEA suggests setting the `MessagesMaximum` value to:

2 * (ack or commit interval) + 1

For example:

If the JMS application acknowledges 50 messages at a time, set the `MessagesMaximum` value to 101.

Tuning the `MessagesMaximum` value too high can cause:

- Increased memory usage on the client.
- Affinity to an existing client as its pipeline fills with messages. For example: If `MessagesMaximum` has a value of 10,000,000, the first consumer client to connect will get all messages that have already arrived at the destination. This condition leaves other consumers without any messages and creates an unnecessary backlog of messages in the first consumer that may cause the system to run out of memory.