# BEA WebLogic® Integration

## Deploying WebLogic Integration Solutions

Version 10.2
Document Revised: March 2008

# Contents

## 1. Introduction



## 2. Configuring a Single-Server Deployment

# 3. Understanding WebLogic Integration Clusters



# 4. Configuring a Clustered Deployment

# 5. Understanding WebLogic Integration High Availability



# 6. Using WebLogic Integration Security

CHAPTER 1

# Introduction

This document describes how to deploy BEA WebLogic® Integration 10.2 solutions in a production environment. The following sections introduce key concepts and tasks for deploying WebLogic Integration (WLI) solutions in your organization:

- Deployment Goals
- Key Deployment Tasks
- Roles in Integration Solution Deployment
- Key Deployment Resources
- Deployment Plans

**Note:** This document focuses on the deployment phase of the software lifecycle for WLI applications. For information about the deployment phase of the software lifecycle for BEA WebLogic Platform applications, see *Deploying Applications to WebLogic Server*.

For examples of source and utilities to build, configure, and deploy WLI applications, see the WLI Solution Samples and the PO Sample.

**Note:** Code samples and utilities are posted on dev2dev for your convenience. They are not supported by BEA.

# Deployment Goals

WLI is a single, unified platform that provides the functionality businesses can use to develop new applications, integrate them with existing systems, streamline business processes, and connect with trading partners.

When deploying WLI solutions, consider the following goals:

- *High Availability.* The deployment must be sufficiently available and accessible, with provisions for failover in the event of hardware or network failures.
- *Performance*. The deployment must deliver sufficient performance at peak and off-peak loads.
- *Scalability.* The deployment must be capable of handling anticipated increases in loads simply by using additional hardware resources, rather than requiring code changes.
- *Security.* The deployment must sufficiently protect data from unauthorized access or tampering.

# Key Deployment Tasks

Deploying WLI may require that you complete some or all of the following tasks:

1. Define the goals for your WLI deployment, as described in "Deployment Goals" on page 1-2.
2. Deploy WLI applications in a cluster. To do so, you must first design the cluster, and before you can start designing, you need to understand the components of a WLI deployment. Chapter 3, "Understanding WebLogic Integration Clusters," provides descriptions of these components to help you design the best possible environment for your application.
3. Deploy WLI applications in a clustered environment so that they are highly available. To do so, you must configure your application as described in Chapter 4, "Configuring a Clustered Deployment."
4. Set up security for your WLI deployment as described in Chapter 6, "Using WebLogic Integration Security."

For a detailed list of deployment tasks associated with WebLogic Platform applications, see *Deploying Applications To WebLogic Server*.

# Roles in Integration Solution Deployment

To deploy an integrated solution successfully, a deployment team must include people who perform the following roles:

- Deployment Specialists
- WebLogic Server Administrators
- Database Administrators

One person can assume multiple roles, and all roles are not equally relevant in all deployment scenarios, but successful deployment requires input from people in each role.

## Deployment Specialists

Deployment specialists coordinate the deployment effort. They are knowledgeable about the features of WLI. They provide expertise in designing the deployment topology for an integration solution, based on their knowledge of how to configure various WLI features on one or more servers. Deployment specialists have experience in the following areas:

- Resource requirements analysis
- Deployment topology design
- Project management

## WebLogic Server Administrators

WebLogic Server administrators provide in-depth technical and operational knowledge about WebLogic Server deployments in an organization. They have knowledge of the hardware and platform, and experience managing all aspects of a WebLogic Server deployment, including installation, configuration, monitoring, security, performance tuning, troubleshooting, and other administrative tasks.

## Database Administrators

Database administrators provide in-depth technical and operational knowledge about database systems deployed in an organization. They have experience in the following areas:

- Hardware and platform knowledge

- Expertise in managing all aspects of a relational database (RDBMS), including installation, configuration, monitoring, security, performance tuning, troubleshooting, and other administrative tasks

# Key Deployment Resources

This section provides an overview of resources that can be modified at deployment time:


- "WebLogic Server Resources" on page 1-4
- "Process Application Resources" on page 1-7
- "Process Control Resources" on page 1-11
- "Message Broker Resources" on page 1-13
- "Event Generator Resources" on page 1-14
- "Trading Partner Integration Resources" on page 1-16
- "Relational Database Management System Resources" on page 1-20
- "Hardware, Operating System, and Network Resources" on page 1-21


**Note:** The term *resource* is used in this document to refer to technical assets in general, except in Chapter 6, "Using WebLogic Integration Security," where it is used to refer only to those underlying WebLogic Server entities that can be protected from unauthorized access using security roles and security policies.

## WebLogic Server Resources

This section provides general information about WebLogic Server resources that are most relevant to the deployment of a WLI solution. You can configure these resources from the WebLogic Server Administration Console or through EJB deployment descriptors.

WebLogic Server provides many configuration options and tunable settings for deploying WLI solutions in any supported environment. The following sections describe the configurable WebLogic Server features that are most relevant to WLI deployments:

- Clustering
- Java Message Service
- EJB Pooling and Caching

- JDBC Connection Pools
- Execution Thread Pool
- J2EE Connector Architecture

## Clustering

To increase workload capacity, you can run WebLogic Server on a cluster: a group of servers that can be managed as a single unit. Clustering provides a deployment platform that is more scalable than a single server.

For more information about clustering, see Chapter 3, "Understanding WebLogic Integration Clusters."

## Java Message Service

The WebLogic Java Message Service (JMS) enables Java applications sharing a messaging system to exchange (create, send, and receive) messages. WebLogic JMS is based on the *Java Message Service Specification* version 1.0.2 from Sun Microsystems, Inc.

JMS servers can be clustered and connection factories can be deployed on multiple instances of WebLogic Server. In addition, JMS event destinations can be configured to handle workflow notifications and messages, as described in "Process Application Resources" on page 1-7.

For more information about WebLogic JMS, see the following:

- Understanding WebLogic JMS in *Programming WebLogic JMS*
- *Configuring and Managing WebLogic JMS*

## EJB Pooling and Caching

In a WLI deployment, the number of EJBs affects system throughput. You can tune the number of EJBs in the system through either the EJB pool or the EJB cache, depending on the type of EJB. The following table describes types of EJBs and their associated tunable parameter.

**Table 1-1 Parameters for Tuning EJBs**

| EJB Type | Tunable Parameter Name | Tunable Parameter Description |
|---|---|---|
| Message-Driven Beans | `max-beans-in-free-pool` | The maximum number of listeners that pull work from a queue. |
| Stateless Session Beans | `max-beans-in-free-pool` | The maximum number of beans available for work requests. |
| Stateful Session Beans | `max-beans-in-cache` | The number of beans that can be active at once. A setting that is too low results in `CacheFullExceptions`. A setting that is too high results in excessive memory consumption. |
| Entity Beans | | |

For more information about controlling throughput by configuring EJBs, see Tuning WebLogic Server EJBs in *WebLogic Server Performance and Tuning*.

## JDBC Connection Pools

Java Database Connectivity (JDBC) enables Java applications to access data stored in SQL databases. To reduce the overhead associated with establishing database connections, WebLogic JDBC provides connection pools that offer ready-to-use pools of connections to a DBMS.

JDBC connection pools are used to optimize DBMS connections. You can tune WLI performance by configuring the size of JDBC connection pools. A setting that is too low results in delays while WLI waits for connections to become available. A setting that is too high results in slower DBMS performance.

For more information about WebLogic JDBC, see *Programming WebLogic JDBC*.

## Execution Thread Pool

The *execution thread pool* controls the number of threads that can execute concurrently on WebLogic Server. A setting that is too low results in sequential processing and possible deadlocks. A setting that is too high results in excessive memory consumption and may cause thrashing.

The number of execution threads also determines the number of threads that read incoming socket messages (socket-reader threads). This number is, by default, one-third the number of execution threads. A number that is too low can result in contention for threads for reading sockets and can sometimes lead to a deadlock.

Set the execution thread pool high enough so that all candidate threads run, but not so high that performance is hampered due to excessive context switching in the system. Monitor your running system to determine empirically the best value for the execution thread pool.

**Note:** Most production applications require an execution thread count greater than the default value. A thread count of 50 is a commonly used value. Be sure to adjust your JDBC connection pool to match your thread count value.

For more information about configuring execution thread pools, see *WebLogic Server Performance and Tuning*.

### J2EE Connector Architecture

The WebLogic J2EE Connector Architecture (JCA) integrates the J2EE Platform with one or more heterogeneous Enterprise Information Systems (EIS). The WebLogic JCA is based on the *J2EE Connector Specification*, Version 1.0, from Sun Microsystems, Inc.

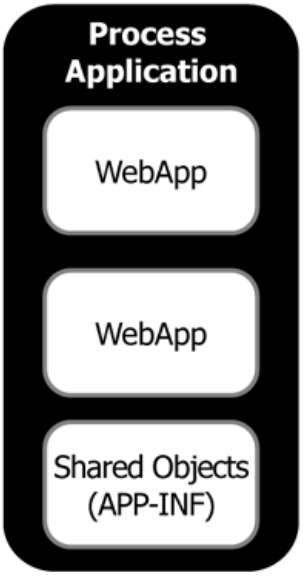
For information about the WebLogic J2EE-CA, see *Programming WebLogic Server Resource Adapters*.

## Process Application Resources

A process application is represented as an EAR file. You compile the EAR file for deployment using the standard procedure for compiling any Workshop application, see *Workshop User's Guide*.

The EAR file consists of multiple web applications and some shared class files. The generated schema files go to the shared class files. Each web application corresponds to a project in the IDE workspace (see Figure 1-1).

**Figure 1-1 Process Application**




Each web application consists of the following items:

- A pool of message-driven beans bound to an input JMS queue

  The input JMS queues use an optimized, internal format for messages. They are used implicitly by WLI and WebLogic Platform components including process controls, Message Broker, buffered messages, and so on.

  **Note:** These input queues are not intended to be used directly by applications. If you are looking for a JMS queue that can be used directly by an application, consider the Workshop SOAP/JMS protocol or the WLI JMS Event Generator.

  For more information about the SOAP/JMS protocol, see *Workshop User's Guide*.

  For more information about the JMS Event Generator, see Event Generators in *Using The WebLogic Integration Administration Console*.

- A pool of message-driven beans bound to an error JMS queue

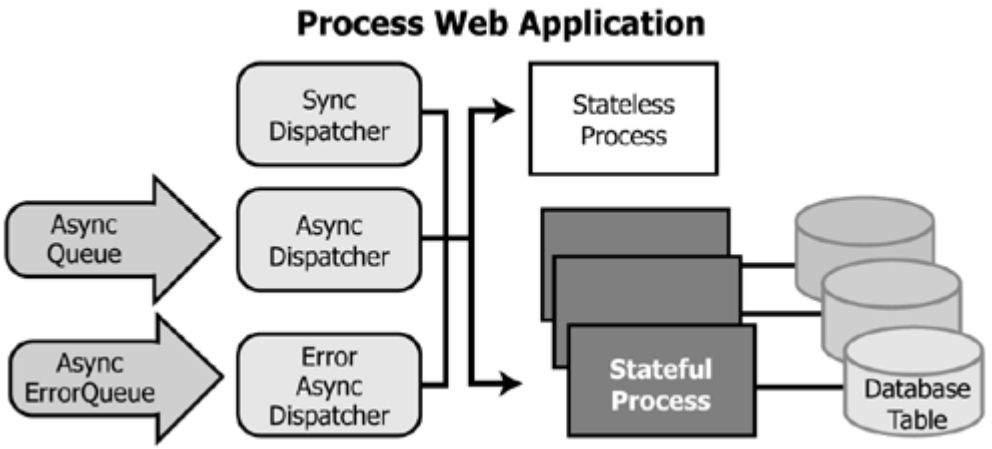
  This is used for dispatching exception elements in process definition.

- A pool of stateless session beans for each stateless process

- A pool of entity beans for each stateful process

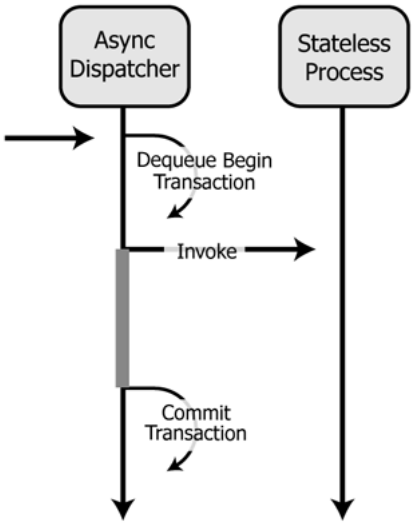
Figure 1-2 shows the components in a process web application.

**Figure 1-2 Process Web Application**



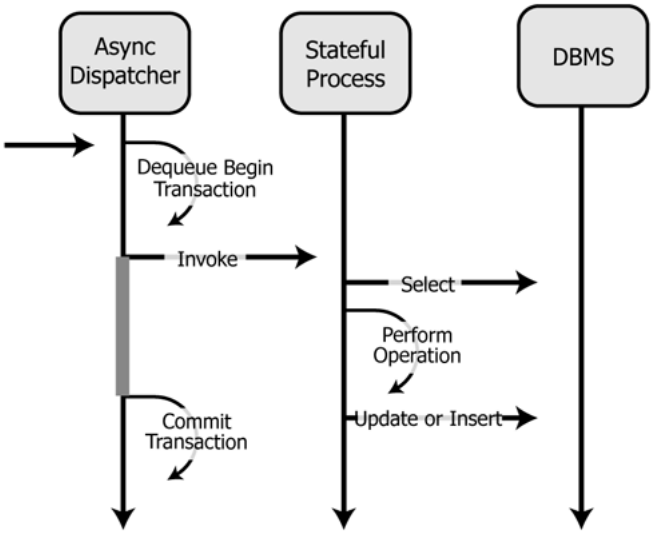

The asynchronous dispatcher has different interactions with stateless and stateful processes. For an illustration of the interaction between the asynchronous dispatcher and a stateless process (see Figure 1-3).

**Figure 1-3 Interaction Between Dispatchers and a Stateless Process**




For an illustration of the interaction between the asynchronous dispatcher and a stateful process, see Figure 1-4.

**Figure 1-4 Interaction Between Dispatchers and a Stateful Process**




# Process Control Resources

The process control allows messages to be sent directly from one process to another, either through RMI or through JMS using an optimized data format. Normal WebLogic Server load balancing rules apply when using RMI or JMS. Typically, the message stays on the same server in a cluster due to server affinity of WebLogic Server load balancing.

An in-memory dispatcher table provides the detailed information needed by the process control to send the message at run time. This dispatcher table is automatically updated when an application is deployed or redeployed.

The behavior of a process call depends on whether it is being used for a synchronous or asynchronous dispatch.

Figure 1-5 shows the behavior of a process control used for a synchronous dispatch.

**Figure 1-5 Process Control Used for a Synchronous Dispatch**

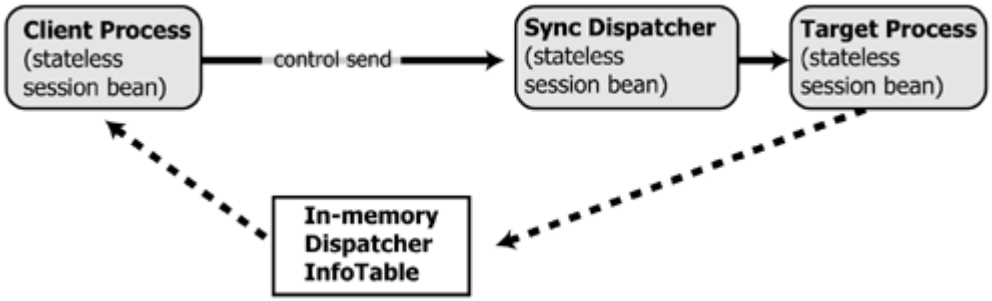
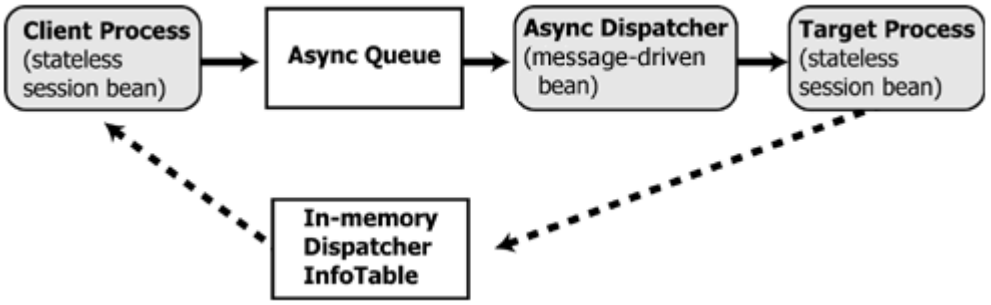



Figure 1-6 shows the behavior of a process control used for an asynchronous dispatch.

**Figure 1-6 Process Control Used for an Asynchronous (Buffered) Call**




It is also possible for a synchronous client process to interact with an asynchronous process.

**Note:** You can optimize performance of this configuration by creating the following dedicated execution thread pools:

- `wli.internal.SyncAsyncTransportServlet`
- `wli.internal.SyncAsyncResponseListener`

Choose a thread pool size that matches the requirements of your application and tracking level.

If you do not create these pools, threads are consumed from the default thread pool.

For more information about this configuration, see Building Synchronous and Asynchronous Business Processes in *Guide to Building Business Processes*.

For implementation examples, see the WLI Solution Samples.

# Message Broker Resources

Any time the Message Broker publishes a message through a Message Broker publish control or event generator, the following actions occur:

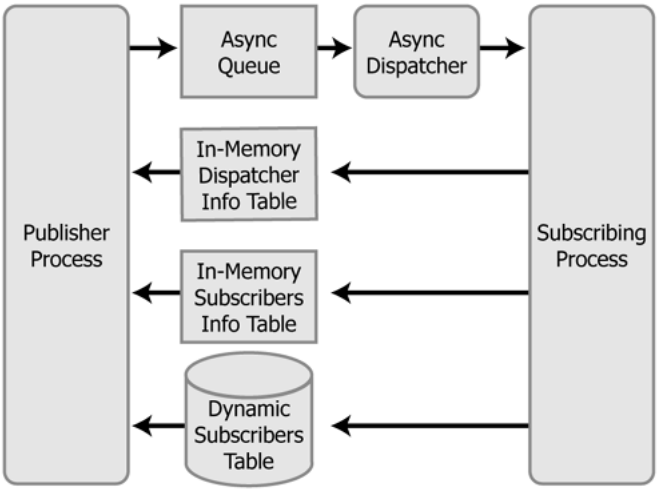
- A list of subscribers is retrieved from the in-memory subscription information table.
- For each *static subscriber* (a subscriber that has a subscription on a start operation) interested in receiving messages listening on a particular channel:
  - If the subscription has no filter, it is sent.
  - If the subscription has a filter, it is evaluated and if it matches, the message is sent.

  **Note:** Database tables are never used for static subscribers.
- For each *dynamic subscriber* (a subscriber that has a subscription on a Message Broker subscriber control) interested in receiving messages listening on a particular channel:
  - The subscriber updates the dynamic subscribers table when doing a subscribe operation.
  - For each unique filter declared on subscription controls, the XQuery declared on the subscription is used to extract an XML fragment from the to-be-published document or metadata. The XML fragment is then effectively used as a select value into the dynamic subscribers table. The result set from the select is then used as the list of subscribers to publish to for that unique filter.

    The Message Broker uses the same JMS asynchronous queue that the process control uses. Once a message is enqueued, it follows the same code path as if it were sent using the process control.

    For more information about publishing using the Message Broker control, see Figure 1-7.

**Figure 1-7 Publishing Using the Message Broker Publish Control**




# Event Generator Resources

WLI has a number of native event generators:

- Email
- File
- HTTP
- JMS
- MQSeries
- RDBMS
- Timer

## JMS Generator

The JMS event generator is packaged as a message-driven bean pool. It can be targeted freely to any number of managed servers in a cluster. It would typically be targeted at either a single managed server (when using a physical JMS destination) or to the cluster (when using distributed destinations).

## File, Email, and Timer Event Generators

These event generators poll for events to happen. To do this, each event generator is packaged as a message-driven bean pool and configured with a specific JMS queue. Messages are sent from the event generator to its associated queue with a delivery time of *poll-interval* in the future. The queue is shared between event generators of the same type (file, email, and so on), and a selector is used to share messages in the queue.

How you target a polling event generator depends on how the JMS server owning the associated queue has been targeted, as shown in Table 1-2.

**Table 1-2 Target a Polling Event Generator**

| Is the JMS target migratable? | Polling event generator target must be a. . . |
|---|---|
| Yes | Cluster |
| No | Single server |

Because the polling event generators would contend with each other during their polls, they are restricted to being active on a single managed server in a cluster. When an event generator is associated with a migratable queue, it is active only on the single server containing the migratable JMS server, even though the polling event generator is targeted at the cluster.

**Note:** WLI supports polling event generators targeted at a single server. This configuration is, however, not appropriate for applications that require high availability.

## MQSeries and HTTP Event Generators

The HTTP event generator is a servlet that takes HTTP requests, checks for the content type, and then publishes the messages to Message Broker channels.

The MQSeries event generator polls for messages on a WebSphere MQ queue and publishes the messages (MQMD headers as metadata along with the message payload) to Message Broker channels. Content filtering, as well as other handling criteria, are specified in the channel rules for the event generator.

Both HTTP and MQSeries event generators can be targeted at a single managed server or cluster.

### RDBMS Event Generator

The RDBMS event generator polls the database table to check for added, deleted, or updated rows and publishes the results to Message Broker channels. You can also use this event generator to run custom queries on the database table and publish the results to Message Broker channels.

For more information, see Event Generators in *Using The WebLogic Integration Administration Console*.

### Suspending Event Generators

The suspended status of an event generator is not preserved when the server is restarted. If the event generator is in the suspended state when the server is restarted, the event generator remains suspended and no events are processed. You can restart the event generator by setting it to 'running' from the WebLogic Integration Administration Console.

For more information, see Event Generators in *Using The WebLogic Integration Administration Console*.

## Trading Partner Integration Resources

Trading Partner Integration (TPI) provides a framework for peer-to-peer business protocols, implementing RosettaNet (versions 1.1 and 2.0) and ebXML (versions 1.0 and 2.0).

**Note:** Trading Partner Integration was formerly known as B2B. Some resource names contain abbreviations that are a legacy from prior WLI releases. The Trading Partner Integration resources currently retain B2B as part of their names

When you deploy WLI to a clustered domain, all Trading Partner Integration resources—with the exception of resources for the administration server—must be deployed homogeneously in the cluster. Targeting Trading Partner Integration resources to all clustered servers in a domain enables you to achieve high availability, scalability, and performance improvements for your application.

For more information about Trading Partner Integration resources and clustering, see "Designing a Clustered Deployment" on page 3-2.

For information about resources that can be configured to accommodate Trading Partner Integration loads, see Trading Partner Management in *Using The WebLogic Integration Administration Console*.

## Trading Partner Management Repository

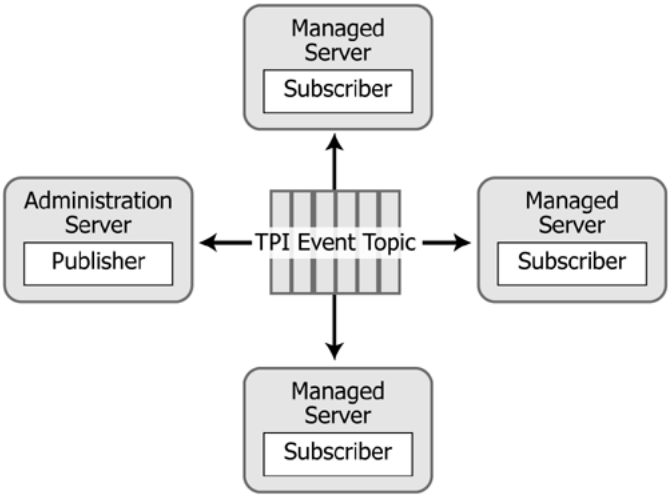
The Trading Partner Management Repository is an important part of Trading Partner Integration. Database operations on this repository and for all of Trading Partner Integration are performed through the `JDBCTxDataSource` named `cgDataSource` using the `JDBCPool` named `cgPool`.

**Note:** You can configure the database for your Trading Partner Management Repository to use concurrent access. For information about issues regarding specific databases, see the *WebLogic Integration 10.2 Release Notes*.

### Data Caching

Data from the Trading Partner Management Repository is cached during server startup to improve performance by reducing access to this resource. In a cluster environment, the Trading Partner Management Repository data is cached on the administration server and each managed server. These caches are synchronized through the mechanism shown in Figure 1-8.

**Figure 1-8 Trading Partner Management Repository Cache Synchronization**




### Managing the Data Cache

The WebLogic Integration Administration Console enables you to perform updates, imports, and deletions to the Trading Partner Management Repository. For information about using the WebLogic Integration Administration Console to perform these operations, see Trading Partner Management in *Using The WebLogic Integration Administration Console*.

## Trading Partner Integration Initialization and Run-Time Operations

Trading Partner Integration is initialized during server startup by the WLI-B2B Startup EJB.

**WARNING:** The WLI-B2B Startup EJB has an `initial-beans-in-pool` setting of 1. Changing this value will cause Trading Partner Integration startup to fail.

At run time, outgoing and incoming Trading Partner Integration messages traverse different paths. The following sections describe the paths and process flows for outgoing and incoming business messages.

### Outgoing Messages

Figure 1-9 shows the path of an outgoing business message.

**Figure 1-9 Outgoing Message Path**

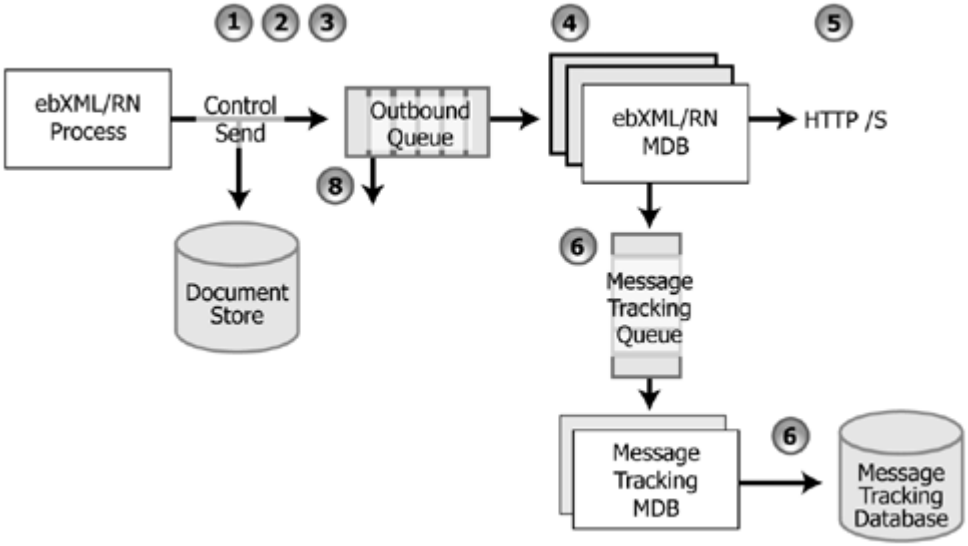



Figure 1-9 illustrates the following process flow:

1. A message is sent from a WLI business process using a Trading Partner Integration control (ebXML or RosettaNet).
2. The Trading Partner Integration layer (RosettaNet or ebXML) uses input from the control (the message, annotations, and so on) and constructs the appropriate message to be sent.

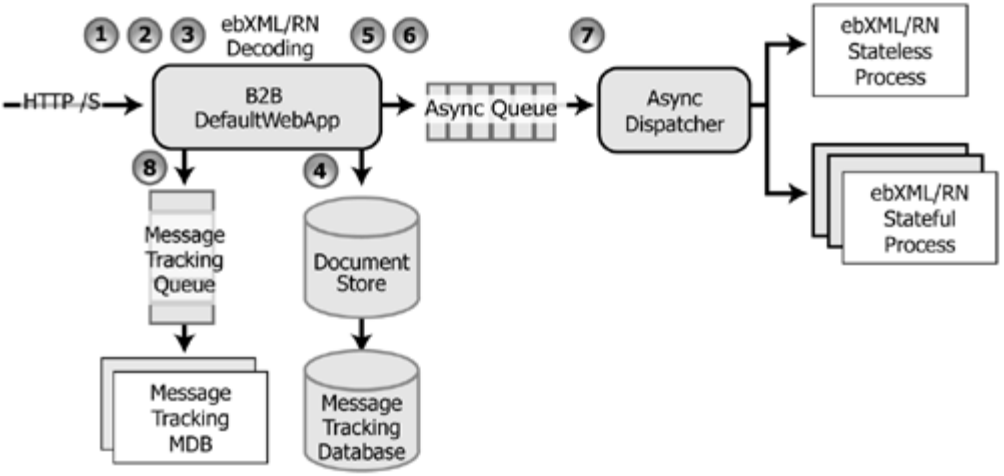
3. The Trading Partner Integration layer persists this message in the WLI document store and forwards it to a JMS queue. RosettaNet and ebXML have their own JMS queues—`wli.internal.b2b.rosettanetencoder.queue` and `wli.internal.b2b.ebxmlencoder.queue`, respectively.
4. Each of these queues has its own message-driven bean(s) listening to the queue. The message-driven beans are WLI-B2B RosettaNet for RosettaNet and WLI-B2B ebXML for ebXML. The pool size of these message-driven beans can be increased as needed to support customer environments that experience high message volume.
5. The message-driven beans send out the message asynchronously over HTTP(S).
6. Message tracking information for outbound messages is sent to the Trading Partner Integration message tracking queue, `wli.internal.msgtracking.queue`. The WLI Message Tracking message-driven bean listens to this queue. It will update the various message tracking tables based on the tracking level set in the Trading Partner Management module of the WebLogic Integration Administration Console.

   For information about using the WebLogic Integration Administration Console to set tracking levels, see Trading Partner Management in *Using The WebLogic Integration Administration Console*.

## Incoming Messages

Figure 1-10 shows the path of an incoming business message.

**Figure 1-10 Incoming Message Path**



The preceding figure illustrates the following process flow:

1. A message sent to a trading partner is received by the TPI Transport Servlet Filter, specified in `B2BdefaultWebApp/WEB-INF/web.xml`. The filter inspects the URL and decides if the incoming request is a TPI URL / request. If it is not destined for Trading Partner Integration, the message continues on to other filters and the final destination servlet.
2. If it is a Trading Partner Integration message, the filter forwards to the Transport Servlet, WLI-B2B HTTP Transport. This servlet is packaged in `b2b.war`.
3. The message is then sent to the Trading Partner Integration decoder. There is a different decoder for each business protocol. The appropriate decoder unpacks the message.
4. The decoder persists the message in WLI Document Store.
5. The decoder determines the destination and originator parties, the service name, and other relevant information that helps in dispatching the message.
6. The message is then dispatched to the Async Dispatcher Queue. If the decoder determines this message is part of a new exchange, a new process instance will be requested. If, however, this message is part of an ongoing exchange, the decoder will request that this be dispatched to a particular receive node within an existing process instance. The message parts will be packaged as appropriate for the receive node's method signature.
7. The Async Dispatcher Module dispatches the message to the appropriate business process.
8. Message tracking information for inbound messages is sent to the Trading Partner Integration Message Tracking queue, `wli.internal.msgtracking.queue`. The WLI Message Tracking message-driven bean listens to this queue. It will update the various message tracking tables based on the tracking level set in the Trading Partner Management module of the WebLogic Integration Administration Console.

   For information about using the WebLogic Integration Administration Console to set tracking levels, see Trading Partner Management in *Using The WebLogic Integration Administration Console.*

# Relational Database Management System Resources

WLI relies extensively on database resources for handling run-time operations and ensuring that application data is durable. Database performance is a key factor in overall WLI performance.

For information about database tuning requirements associated with WLI applications, see Preparing Your Database and the database-specific notes in Maintaining Availability.

For additional information on turning your database, see your database vendor's documentation.

## Hardware, Operating System, and Network Resources

Hardware, operating system, and network resources play a crucial role in WLI performance. Deployments must comply with the hardware and software requirements described in the *WebLogic Integration 10.2 Release Notes*.

# Deployment Plans

You can use deployment plans to configure WLI applications for deployment.

Deployment plans are optional XML documents that you can use to configure WLI applications for deployment to specific WebLogic Server environments. By using deployment plans, you can override property values that are defined in deployment descriptors, or define property values that are not explicitly defined in the deployment descriptors.

For more information, see the following sections in Configuring Applications for Production Deployment:

- Creating a New Deployment Plan to Configure an Application: This section describes the procedure to create deployment plans.
- Understanding Deployment Plan Contents: This section contains a sample deployment plan.

You can also use the PlanGenerator tool to export portions of the deployment configuration to a deployment plan. For more information, see weblogic.PlanGenerator Command Line Reference.

CHAPTER 2

# Configuring a Single-Server Deployment

This section describes the tasks that you must perform to configure WLI for deployment in a single WebLogic Server environment.

To set up and deploy WLI in a single-server configuration, complete the following steps:

- Step 1. Configure a Database
- Step 2. Prepare the WLI Domain
- Step 3. Configure WebLogic Integration Security
- Step 4. Start and Monitor the Server
- Step 5. Deploy a WLI Application

## Step 1. Configure a Database

Configure one of the following databases for your domain:

- IBM DB2
- Microsoft SQL Server
- Oracle
- Sybase

**Notes:** It is important to configure your database appropriately for production use. You must provide adequate space to store data and log messages, and follow best practices for administering your database.

You can configure your database to use concurrent access.

For information about database tuning requirements associated with WLI applications, see Preparing Your Database and the database-specific notes in Maintaining Availability.

For the latest information about specific databases, see the *WebLogic Integration 10.2 Release Notes*.

# Step 2. Prepare the WLI Domain

You begin the definition of a WLI deployment by creating a domain using the BEA Domain Configuration Wizard.

**Note:** The procedure described in this section for setting up your domain is based on the assumption that you are running the Domain Configuration Wizard in GUI mode from the Windows Start menu. For information about using the Domain Configuration Wizard in different modes, see *Creating WebLogic Domains Using the Configuration Wizard*.

To create a WLI domain using the Domain Configuration Wizard, complete the following steps:

1. From the Start Menu, choose **All Programs** > **BEA Products** > **Tools** > **Configuration Wizard**.

   The **Welcome** page of the **BEA WebLogic Configuration Wizard** appears.

2. Select **Create a new WebLogic domain** and click **Next**. The **Select Domain Source** page appears.

   As you proceed through the Configuration Wizard, several pages will appear in a sequence. You need to specify your settings on each page and click **Next** to proceed to the subsequent page.Table 2-1 lists the pages and the options that you need to select to create the domain successfully.

**Table 2-1 Configuring the Domain Using the Configuration Wizard**

| Page in the Configuration Wizard Dialog Box | Recommendation Action |
|---|---|
| Select Domain Source | Select **Generate a domain configured automatically to support the following BEA Products** option for the following BEA products:<br>• WebLogic Server (Required)<br>• WebLogic Integration |
| Configure Administrator Username and Password | Specify the following mandatory credentials:<br>User name = `weblogic`<br>User password = `weblogic`<br>Confirm user password = `weblogic` |
| Configure Server Start Node and JDK | Select the appropriate mode (development or production) in the **WebLogic Domain Startup Mode** column.<br>Select the appropriate JDK in the **JDK Selection** column. |
| Customize Environment and Service Settings | Select **Yes**, to configure the settings defined in the domain source. |
| Configure the Administration Server | Accept all the default values.<br>**Note:** When you configure the administration server, we recommend that you accept the default Server Name (`AdminServer`), as prompted by the Domain Configuration Wizard. |
| Configure Managed Servers | If you want to distribute your WebLogic configuration across managed servers, click **Add**, enter a name and the required configuration details. |
| Configure Clusters | If you want to distribute your WebLogic configuration across managed clusters, click **Add**, enter a name and the required configuration details. |
| Configure Machines | If you want to distribute your WebLogic configuration across physical machines, click **Add**, enter a name and the required configuration details. |
| Configure JDBC Data Sources | Accept the defaults for `cgDataSource` and `bpmArchDataSource`. Add any application-specific data sources.<br>**Note:** No more than one non-XA datasource can be used in the same transaction. Data sources cannot share an XA JDBC connection pool. |

**Table 2-1 Configuring the Domain Using the Configuration Wizard**

| Page in the Configuration Wizard Dialog Box | Recommendation Action |
|---|---|
| Run Database Scripts | Click on **cgDataSource-nonXA** in the **Available JDBC Database Scripts**, select all the SQL scripts from the **Available SQL Files and Database Loading Options** and click **Run Scripts**.<br>Perform the above procedure for **p13nDataSource** in the **Available JDBC Database Scripts**.<br>**Note:** When preparing a production environment (working in *noniterativedev* mode), you can create the WLI database tables. For the procedure to create these tables, see "Creating the WebLogic Integration Tables" in Configuring a Production Database. |
| Configure JMS File Stores | Accept the defaults and add any file stores needed by the application. |
| Review WebLogic Domain | Review the selections you made so far. |
| Create WebLogic Domain | On the Create WebLogic Domain page, specify the following values for each field and click **Create**:<br>• **Domain name:** Give a domain name<br>• **Domain Location:** `<Base Directory>\bea\user_projects\domains`<br>• **Application Location:** `<Base Directory>\bea\user_projects\applications`<br>After the domain is created successfully, the Creating Domain page is displayed<br>Select the **Start Admin Server** check box and click **Done** to proceed. |

When you complete the domain configuration using the Domain Configuration Wizard, your new domain is created in the location you specified.

Your WLI domain includes two configuration files:

- `config.xml` contains a definition for the administration server.
- `wli-config.properties` contains other domain-specific information which WLI uses.

For more information about `config.xml`, see *Understanding Domain Configuration*.

For information about `wli-config.properties`, see Appendix A, "wli-config.properties Configuration File."

# Step 3. Configure WebLogic Integration Security

If you want to configure SSL for your domain, you can do so by using the WebLogic Server Administration Console. For information about the tasks you must complete, see:

- Configuring SSL.
- Chapter 6, "Using WebLogic Integration Security."

For general information about configuring security for WebLogic Platform applications, see Security.

# Step 4. Start and Monitor the Server

This section describes how to start, monitor, and shut down the server in your WLI domain:

- Starting the Server
- Monitoring and Shutting Down Your Server

For information about starting servers for WebLogic Platform applications, see Starting and Stopping Servers: Quick Reference.

## Starting the Server

To start the server, complete the following procedure:

From the Start menu, choose **All Programs > BEA Products > User Projects > Domain Name > Admin Server Console**.

For more information about starting the server, see Starting and Stopping Servers: Quick Reference.

## Monitoring and Shutting Down Your Server

Once startup is complete, you can use the WebLogic Server Administration Console to verify deployments and status. For information about using WebLogic Server Administration Console to monitor your server, see Monitor Servers.

If you need to shut down your WLI application, use the WebLogic Server Administration Console.

**Note:** It is recommended that you do not close the command window or press Ctrl+c to stop WLI.

For the procedure to shut down your application gracefully, see Control graceful shutdowns.

# Step 5. Deploy a WLI Application

Once you have configured and secured your WLI domain and added the queues and database tables for your application to that domain, you can use the WebLogic Server Administration Console to deploy the EAR file that contains your WLI application.

If you did not configure all the queues necessary for your application while configuring your WLI domain, you can configure them now using the WebLogic Server Administration Console.

**Note:** Async request and async request error queues, as well as conversational state database tables, are created automatically for applications in the development environment.

In the production environment, however, you must create the queues and tables manually. For more information, see the following:

- "Creating Conversational State Database Tables" in *Configuring the Production Database*.
- "Adding Application Resources Required by the WebLogic Workshop Runtime" in *Creating and Configuring the WebLogic Domain*.

**Note:** If your WLI solution uses the RDBMS Event Generator, be sure to configure the Redelivery Delay Override setting appropriately for `wli.internal.egrdbms.queue`. For the procedure to configure the Redelivery Delay Override, see "RDBMS Event Generator" on page 3-10.

For the procedure to deploy an EAR file, see Enterprise Applications in the WebLogic Server Administration Console Online Help.

For information about deploying through the command line, see *weblogic.Deployer Command-Line Reference*.

For examples of automation scripts that build, configure, and deploy WLI applications outside of an interactive console environment, see the WLI Solution Samples and the PO Sample.

**Note:** Code samples and utilities are posted on dev2dev for your convenience. They are not products supported by BEA.

**Note:** As your business requirements change, you may need to develop and deploy new versions of your WLI application. For more information see, *Deploying Applications to WebLogic Server*.

CHAPTER 3

# Understanding WebLogic Integration Clusters

The following sections describe how WebLogic Integration (WLI) is configured and deployed in a clustered environment. It contains the following topics:

- Understanding WLI Clusters
- Designing a Clustered Deployment
- Load Balancing in a WLI Cluster
- Deploying Event Generators

For more information about deploying applications, see Deploying Applications To WebLogic Server.

## Understanding WLI Clusters

Clustering allows WLI to run on a group of servers that can be managed as a single unit. In a clustered environment, multiple machines share the processing load. WLI provides load balancing so that resource requests are distributed proportionately across all machines. A WLI deployment can use clustering and load balancing to improve scalability by distributing the workload across nodes. Clustering provides a deployment platform that is more scalable than a single server.

A WLI cluster domain consists of only one administration server, and one or more managed servers. The managed servers in a WLI domain can be grouped in a cluster. When you configure WLI clusterable resources, you normally target the resources to a named cluster. The advantage of specifying a cluster as the target for resource deployment is that it makes it possible to increase capacity dynamically by adding managed servers to your cluster.

**Note:** WLI domains can support multiple clusters. However, you must still target WLI system resources and applications on a single WLI cluster in a multiple cluster domain.

The topics in this section provide the information you need to configure WLI in a clustered environment. Although some background information about how WLI supports clustering is provided, the focus is on procedures that are specific to configuring WLI for a clustered environment.

Before proceeding, we recommend that you review the following sections of the WebLogic Server documentation to obtain a more in-depth understanding of clustering:

- *Using WebLogic Server Clusters*
- "Using WebLogic Server Clusters to Improve Performance" in *Top Tuning Recommendations for WebLogic Server.*

# Designing a Clustered Deployment

The following sections provide the information you need to design a clustered deployment:

- Introducing WLI Domains
- Deploying WLI Resources
- Load Balancing in a WLI Cluster

## Introducing WLI Domains

Before you begin designing the architecture for your clustered domain, you need to learn how WebLogic Server clusters operate.

### Creating Domains

Domain and cluster creation are simplified by a Domain Configuration Wizard that lets you generate domains from basic and extension domain templates. Based on responses to user queries, the Domain Configuration Wizard generates a domain, server, and enterprise application

with the appropriate components preconfigured and assets included. For information about the templates available for different domains, see the Template References.

The domain you create with the Configuration Wizard must have one and only one target which will contain WLI system components and applications. You can specify this target by setting the value of `weblogic.wli.WliClusterName` in `wli-config.properties`. If you do not specify a value for `weblogic.wli.WliClusterName`, the WLI target defaults to the first WLI cluster. For more information about the `wli-config.properties` file, see Appendix A, "wli-config.properties Configuration File."

### Clustered Servers

A server can be either a managed server or an administration server. A WebLogic Server running the administration service is called an administration server and hosts the WebLogic Server Administration Console. In a domain with multiple WebLogic Servers, only one server is the administration server; the other servers are called managed servers. Each managed server obtains its configuration at startup from the administration server.

For more information about WebLogic clusters, see *Using WebLogic Server Clusters* in the WebLogic Server documentation set. This document includes details regarding recommended basic, multi-tiered, and proxy architectures. For information about security considerations in the design of WebLogic clusters, see "Security Options for Cluster Architectures" in Cluster Architectures in *Using WebLogic Server Clusters*.

All the managed servers in a WebLogic Integration domain must be a part of the WebLogic Integration cluster, you cannot have a stand alone managed servers.

### Note About Cluster and Management Domains

Although it is possible for a WebLogic Server management domain and cluster domain to be different (that is, it is possible for WebLogic Server clusters to have nodes that belong to different management domains), you must design your WLI deployment such that the cluster domain equals the management and security domain.

## Deploying WLI Resources

For each server in a clustered domain, you can configure a variety of attributes that define the functionality of the server in the domain. These attributes are configured automatically when you create a WLI domain using the Configuration Wizard. You can also configure these attributes manually using the Servers node in the WebLogic Server Administration Console.

**Note:** The WLI applications must be deployed to the entire cluster; they cannot be deployed to just a few managed servers in the cluster.

For a list of configurable WLI deployment resources, see Appendix B, "WebLogic Integration Deployment Resources." It describes the default targeting of each resource in a clustered WLI domain and provides instructions on how to navigate to each resource in the WebLogic Server Administration Console.

This section contains the following topics regarding additional WLI deployment configuration requirements:

- Two-Phase Deployment of WLI
- Trading Partner Integration Resource Configuration
- Cluster Configuration Changes and Deployment Requests

## Two-Phase Deployment of WLI

It is essential to have all WLI application components deployed before your system attempts to process messages. To guarantee this, specify the `TwoPhase` attribute when you deploy WLI. The following excerpt from a sample `config.xml` file illustrates an Application element that specifies the two-phase deployment of WLI.

**Listing 3-1 Deploying the WLI Application**

```
<Domain Name="MyCluster">
...
    <Application Name="WebLogic Integration" Path="WL_HOME/lib"
TwoPhase="true">
...
```

## Trading Partner Integration Resource Configuration

Trading Partner Integration components must be deployed homogeneously to a cluster. You must configure Trading Partner Integration resources identically on every managed server so that there is no single point of failure.

When configuring Trading Partner Integration in a cluster, keep in mind the following considerations:

- The HTTP/HTTPS endpoints you specify in the bindings of trading partners must be the host and port number of the hardware or software router. This protects the identity of your managed servers (which are normally behind a firewall), and allows managed servers to change operational status without impacting the external customer.
- You perform Trading Partner Management configuration updates through the WebLogic Integration Administration Console. A JMS broadcast mechanism propagates these changes to the managed servers. Changes take place quickly, but not instantaneously. There is a brief window during which the managed servers have a mix of old and new configuration information. You can minimize the impact of these changes on users by performing changes when the resources you are updating are inactive.
- Note that the ebXML and RN protocols are very nearly stateless. As a result, multiple messages within the same conversation are normally processed by different nodes in the cluster.

### Cluster Configuration Changes and Deployment Requests

You can only change configuration for a cluster (for example, add new nodes to the cluster or modify Trading Partner Integration configuration) while its administration server is active.

If the administration server for a cluster is down, deployment or undeployment requests are interrupted, but managed servers continue serving requests. You can boot or reboot managed servers using an existing configuration, as long as the required configuration files (`msi-config.xml`, `SerializedSystemIni.dat`, and optionally `boot.properties`) exist in each managed server's root directory.

Managed servers that start without an administrative server operate in Managed Server Independence (MSI) mode. For complete information about MSI mode, see "Managed Server Independence Mode" in Managing Server Startup And ShutDown.

## Load Balancing in a WLI Cluster

One of the goals of clustering your WLI application is to achieve scalability. In order for a cluster to be scalable, each server must be fully utilized. Load balancing distributes the workload proportionally among all the servers in a cluster so that each server can run at full capacity. The following sections describe load balancing for various functional areas in a WLI cluster:

- Load Balancing HTTP Functions in a Cluster
- Load Balancing JMS Functions in a Cluster

For more information, see Load Balancing in a Cluster in *Using WebLogic Server Clusters*.

## Load Balancing HTTP Functions in a Cluster

Both Web services (SOAP or XML over HTTP) and WebLogic Trading Partner Integration protocols can use HTTP load balancing. External load balancing can be accomplished through the WebLogic HttpClusterServlet, a WebServer plugin, or a hardware router.

WebLogic Server supports load balancing for HTTP session states and clustered objects. For more information, see Communications in a Cluster in *Using WebLogic Server Clusters*.

## Load Balancing JMS Functions in a Cluster

Most JMS queues used by WLI or WLI applications are configured as distributed destinations. The exceptional cases are JMS queues that are targeted to single managed servers.

For detailed information on JMS load balancing, see "Tuning Distributed Destinations" in Tuning WebLogic JMS.

### Load Balancing for Synchronous Clients and Asynchronous Business Processes

If your WLI solution includes communication between a synchronous client and an asynchronous business process, the `weblogic.jws.jms.QueueConnectionFactory` must have server affinity enabled. This is the default setting.

**WARNING:** Attempting to tune JMS load balancing by disabling server affinity for a solution that includes communication between a synchronous client and an asynchronous business process will result in unpredictable behavior.

### Load Balancing for RDBMS Event Generators

The RDBMS Event Generator has a dedicated JMS connection factory (`wli.internal.egrdbms.XAQueueConnectionFactory`). Load balancing is enabled for this connection factory by default. To disable load balancing for RDBMS events, you must disable load balancing and enable server affinity for `wli.internal.egrdbms.XAQueueConnectionFactory`.

# Deploying Applications

Applications are deployed in production after creating EAR files from a Workshop application. Deploying a process application uses the same steps as deploying a web service application.

When deploying a WLI application to a cluster, keep in mind the following considerations:

- The WebLogic Integration applications must be targeted to the entire cluster.
- When deploying to a cluster, we recommend that the queues referred to in `wlw-manifest.xml` be configured as JMS Distributed Queues, with a member of the distributed queue configured on each managed server.
- You need to create database tables for each conversational state element referred to in `wlw-manifest.xml`.
- When using the process control to communicate between processes, a target process must be deployed on the same managed server as the client process. If the target process is not deployed on the same server as the client process, the dispatching table will not be updated and the client process will lack the necessary dispatching information to call the target process.
- Like communication between processes using the process control, subscriber processes must be deployed on the same server as the publisher.

For a full overview of application deployment, see *Deploying Applications To WebLogic Server*.

# Deploying Event Generators

WLI event generators (Email, File, HTTP, JMS, MQSeries, RDBMS, and Timer) can be deployed through the WebLogic Integration Administration Console. For information about how to deploy event generators using the WebLogic Integration Administration Console, see "Creating and Deploying Event Generators" in Event Generators in *Using The WebLogic Integration Administration Console.*

Event generators can also be deployed using the WebLogic Scripting Tool (WLST). The following three pieces of WLST script show how to create, deploy, and configure an event generator.

**Note:** WLST Offline and WLST Online are available for download and evaluation form BEA's dev2dev site, but have not been formally included in the WebLogic Platform 10.2 product. WLST is supported through BEA newsgroups only, and the utility and APIs are subject to change. BEA intends to formally support this capability in a future release of WebLogic Platform.

The following excerpt from a WLST script shows how to create a JMS event generator:

**Listing 3-2 Creating an Event Generator Using WLST**

```
import com.bea.wli.mbconnector.jms as eggen
…
eggen.JmsConnGenerator.main([
       "-inName", “myEgName”,
       "-outfile", “mydomain/myEgName.jar",
       "-destJNDIName", “myQueueName”,])
```

Once you have created the event generator, you can deploy it to the appropriate target using script similar to the following example:

**Listing 3-3 Deploying an Event Generator Using WLST**

```
wlst.deploy( "WLIJmsEG_myEgName”, “mydomain/myEgName.jar", myServer )
```

To configure the properties of the deployed event generator, use a script similar to the following example:

**Listing 3-4 Configuring an Event Generator Using WLST**

```
import com.bea.wli.management.configuration as wlicfg
…
# Must have wli.jar in classpath
egCfgMBean = wlst.getTarget("JMSEventGenerators/JMSEventGenerators")
egMBean = egCfgMBean.newJMSEventGenConfigurationMBean( “myEgName”)
cData = jarray.zeros( 1, wlicfg.JMSEventGenChannelConfiguration )
cData[0] = wlicfg.JMSEventGenChannelConfiguration()
cData[0].setChannel( “myChannel” )
cData[0].setComment("Default channel")
```

```
egMBean.setChannels(cData);
```

For more information about automating deployment with WLST and other utilities, see "Automating the Promotion Progress" in Overview of WebLogic Platform Deployment in *Deploying WebLogic Platform Applications*.

**Note:** Code samples and utilities are posted on dev2dev for your convenience. They are not products supported by BEA.

The following sections provide additional guidelines for event generators.

# Email, File, and Timer Event Generators

The email, file, and timer event generators should be targeted to the cluster. They will be active on the managed server containing the migratable server with the queues associated with the specific event generator (for example, `wli.internal.egmail.queue` for an email event generator).

# JMS Event Generator

The sections below describe targeting and error handling issues to take into consideration when you deploy the JMS event generator.

## JMS Event Generator Targeting

The JMS event generator should be targeted depending on the destination JNDI name of the JMS event generator as indicated in the following table:

**Table 3-1 JMS Event Generator Targeting**

| If the JMS destination is a. . . | Target the. . . |
|---|---|
| Distributed destination | Cluster |
| Destination on a migratable server | Cluster<br>**Note:** The event generator will only be active on the managed server that currently hosts the destination. |
| Destination on a non-migratable server | Managed server with the destination. |

### JMS Event Generator Error Handling

The JMS event generator has no explicit error handling mechanism. Error handling is provided by associating the JMS event generator queue with an error queue.

**Note:** It is important to set the Redelivery Limit for a JMS error queue to a number of messages that is practical for your environment. By default, a JMS queue will redeliver error messages and warnings an infinite number of times.

For information about how to set the Redelivery Limit for messages, see JMS > Queue > Redelivery in the WebLogic Server Administration Console Online Help.

## MQSeries Event Generator

The MQSeries event generator polls for messages on a WebSphere MQ queue and publishes the messages (MQMD headers as metadata along with the message payload) to Message Broker channels. Content filtering, as well as other handling criteria, are specified in the channel rules for the event generator.

## HTTP Event Generator

The HTTP event generator is a servlet, which takes HTTP requests, checks for the content type, and then publishes the messages to Message Broker channels.

## RDBMS Event Generator

The RDBMS event generator polls the database table to check for added, deleted, or updated rows and publishes the results to Message Broker channels. You can also use this event generator to run custom queries on the database table and publish the results to Message Broker channels.

When deploying the RDBMS event generator in a cluster, you need to manually set the Redelivery Delay Override value to `20000` (20 seconds) and the Redelivery Limit to `-1` (indicating no limit) for each of the RDBMS event generator JMS queues. You must configure these redelivery settings before generating any events on the cluster.

**WARNING:** Leaving the Redelivery Delay Override and Redelivery Limit set to their default values causes two immediate redelivery attempts for a JMS message when an error condition is encountered. If the second redelivery attempt fails, the message is discarded.

To configure the redelivery settings for the RDBMS event generator JMS queues in a cluster, complete the following procedure:

1. If you have not done so already, start the WebLogic Server Administration Console.

   For the procedure to start the WebLogic Server Administration Console (and the administration server, if necessary), see Start the Console.

2. In the left panel of the WebLogic Server Administration Console, navigate to **Domain > Services > JMS > Distributed Destinations > dist_wli.internal.egrdbms.queue_auto > Members**.

3. For each JMS queue listed (`dist_wli.internal.egrdbms.queue_auto_1` through *n*), select the Redelivery tab, and then enter the following values:

   – `20000` in the Redelivery Delay Override field

   – `-1` in the Redelivery Limit field

You must also set the Redelivery Delay Override value to `20000` (20 seconds) for single-server deployments. To configure the Redelivery Delay Override value for single-server deployments, complete the following procedure:

1. If you have not done so already, start the WebLogic Server Administration Console.

   For the procedure to start the WebLogic Server Administration Console (and the administration server, if necessary), see Start the Console.

2. In the left panel of the WebLogic Server Administration Console, navigate to **Domain > Services > JMS > Distributed Destinations > wli.internal.egrdbms.queue**.

3. Select the **Redelivery** tab, and then enter `20000` in the **Redelivery Delay Override** field.

It is not necessary to manually configure the Redelivery Limit setting for single-server deployments. The Redelivery Limit setting defaults to the correct values in single-server deployments.

CHAPTER 4

# Configuring a Clustered Deployment

This section describes the tasks that you must perform to configure WebLogic Integration (WLI) for deployment in a clustered environment.

After planning the architecture of your clustered domain, as described in "Designing a Clustered Deployment" on page 3-2, you are ready to set up WLI in a clustered environment. To do this, you must configure a router (hardware or software), an administration server, and managed servers, and then deploy WLI resources to the servers. The persistent configuration for a domain of WebLogic Server instances and clusters is stored in an XML configuration file (`config.xml`) on the administration server.

To set up and deploy WLI in a clustered domain, complete the following steps:

- Step 1. Comply with Configuration Prerequisites
- Step 2. Prepare the WLI Domain
- Step 3. Configure WebLogic Integration Security
- Step 4. Start and Monitor the Managed Servers in the Domain
- Step 5. Deploy a WLI Application

For a complete list of tools available to automate the application deployment process, see "Automating the Promotion Process" in Overview of WebLogic Platform Development in *Deploying WebLogic Platform Applications*.

For information about deploying WLI on a single server, see Chapter 2, "Configuring a Single-Server Deployment." For a detailed list of deployment tasks associated with WebLogic

Platform applications in general, see Deployment Checklist in *Deploying WebLogic Platform Applications*.

# Step 1. Comply with Configuration Prerequisites

This section describes prerequisites for configuring WebLogic Integration to run in a clustered environment:

- Obtain a WebLogic Server cluster license for each required installation.

  To use WebLogic Server in a clustered configuration, you must have a special cluster license. Contact your BEA representative for information about obtaining one.

- Obtain an IP address for the administration server you will use for the cluster.

  All WebLogic Server instances in a cluster use the same administration server for configuration and monitoring. When you add servers to a cluster, you must specify the administration server that each will use.

- Define IP addresses for the servers in your cluster. You can do this in a number of ways:

  **Note:** You are prompted to provide listen addresses for servers when you create a WLI domain using the Domain Configuration Wizard. (See "Step 2. Prepare the WLI Domain" on page 4-5.)

  – Assign a single IP address and different listen port numbers to the servers in the cluster.

    By assigning a single IP address for your clustered servers with a different port number for each server, you can set up a clustered environment on a single machine without the need to make your machine a multihomed server.

To access such an IP address from a client, structure the IP address and port number in your URL in one of the following ways:

| | |
|---|---|
| *ipAddress:portNumber-portNumber* | When the port numbers are sequential, for example:<br>`127.0.0.1:7003-7005` |
| *ipAddress:portNumber+...+portNumber* | When the port numbers are not sequential, for example:<br>`127.0.0.1:7003+7006+7008` |
| *ipAddress:portNumber,ipAddress:portNumber,...* | Verbose, explicit specification, for example:<br>`127.0.0.1:7003,127.0.0.1:7004,127.0.0.1:7005` |

– Assign a static IP address for each WebLogic Server instance to be started on each machine in the cluster.

  In this case, when multiple servers are run on a single machine, that machine must be configured as a multihomed server, that is, multiple IP addresses are assigned to a single computer. Under these circumstances, you structure the cluster address as a comma-separated list of IP addresses.

  For example, the following listing is an example of a cluster address specified in a `config.xml` file. It specifies a static IP address for each of the four servers in a cluster named `MyCluster`:

```
<Cluster
ClusterAddress="127.0.0.1:7001,127.0.0.2:7001,127.0.0.3,127.0.0.4:7001" Name="MyCluster"/>
```

  You can also use a DNS approach to identifying servers.

  For more information on addressing issues, see "Avoiding Listen Address Problems" in Setting Up WebLogic Clusters in *Using WebLogic Server Clusters*.

**Note:** In test environments, it is possible to have multiple WebLogic Server instances on a single machine. In these circumstances, you can have some WebLogic Server instances on the same node with different port numbers and some on different nodes with the same port number.

- If your WLI domain contains multiple clusters, specify the name of the WLI target in `wli-config.properties` by uncommenting the `weblogic.wli.WliClusterName` property and setting it to the name of the cluster as used in the WebLogic Server Administration Console.

For example, the following is an example of a WLI target, `wliCluster`, specified in a `wli-config.properties` file:

```
weblogic.wli.WliClusterName=wliCluster
```

- Configure one of the following databases for your clustered domain:
  - IBM DB2
  - Microsoft SQL Server
  - Oracle
  - Sybase

  It is important to configure your database appropriately for production use. You must provide adequate space to store data and log messages, and follow best practices for administering your database.

  **Note:** You can configure your database to use concurrent access.

  For detailed information regarding configuring databases for WebLogic Platform applications, see Creating and Configuring the Production Database in *Deploying WebLogic Platform Applications*.

  For information about database tuning requirements associated with WLI applications, see Preparing Your Database and the database-specific notes in Maintaining Availability.

  For the latest information about issues regarding specific databases, see the *WebLogic Integration 10.2 Release Notes*.

- Include a shared file system. A shared file system is required for any cluster you want to be highly available. We recommend either a Storage Area Network (SAN) or a multiported disk system.

  For information about configuring a highly available cluster, see *Programming WebLogic JMS*.

- Configure a hardware or software router for your system. Load balancing can be accomplished using either the built-in load balancing capabilities of a WebLogic proxy plug-in or separate load balancing hardware.

  For more information about load balancing for WebLogic Platform applications, see "Configuring Load Balancing and Failover in a Cluster" in Creating and Configuring the WebLogic Domain in *Deploying WebLogic Platform Applications*.

  For information about hardware and software routers, see *Using WebLogic Server Clusters*.

**Note:** Additional requirements apply when you design your domain to include one or more firewalls. For a description of how to add firewall information to your domain configuration file, see "Adding Proxy Server or Firewall Information to Domain Configuration." For more information, see Communications in a Cluster in *Using WebLogic Server Clusters*.

For more information about setting up clustered WebLogic Server instances, see Setting Up WebLogic Clusters in *Using WebLogic Server Clusters*.

# Step 2. Prepare the WLI Domain

You begin the definition of a WLI deployment by creating a domain using the BEA Domain Configuration Wizard.

**Note:** The procedure described in this section for setting up your domain is based on the assumption that you are running the Domain Configuration Wizard in GUI mode from the Windows Start menu. For information about using the Domain Configuration Wizard in different modes, see *Creating WebLogic Domains Using the Configuration Wizard*.

To create a WLI domain using the Domain Configuration Wizard, complete the following steps:

1. From the Start Menu, choose **All Programs** > **BEA Products** > **Tools** > **Configuration Wizard**.

   The **Welcome** page of the **BEA WebLogic Configuration Wizard** appears.

2. Select **Create a new WebLogic domain** and click **Next**. The **Select Domain Source** page appears.

   As you proceed through the Configuration Wizard, several pages will appear in a sequence. You need to specify your settings on each page and click **Next** to proceed to the subsequent page.Table 4-1 lists the pages and the options that you need to select to create the domain successfully.

**Table 4-1 Configuring the Domain Using the Configuration Wizard**

| Page in the Configuration Wizard Dialog Box | Action |
|---|---|
| Select Domain Source | Select **Generate a domain configured automatically to support the following BEA Products**, and select the following BEA products:<br>• WebLogic Server (Required)<br>• WebLogic Integration |
| Configure Administrator Username and Password | Specify the username and password. |
| Configure Server Start Mode and Java SDK | Select **Production Mode**, and then select either the Sun SDK or JRockit SDK.<br>**Note:** When running WebLogic Integration in a cluster with JRockit, the JVM may report a `Stack Overflow`. If not addressed, the problem can eventually result in a JVM core dump. To prevent the stack overflow issue, you must set the `Thread Stack Size` parameter appropriately. For more information about this parameter, see "Setting the Thread Stack Size" in Tuning WebLogic JRockit JVM.<br>If the thread stack size has not been set, the default value depends on the threading system and the platform on which WebLogic JRockit is running:<br>• **32-bit Default**<br>On either Windows or Linux IA32 machines, the default thread stack size values for native threads are:<br>Win32: 64 kB<br>Linux32: 128 kB<br>• **64-bit Default**<br>On either Windows or Linux IA64 machines, the default thread stack size values for native threads are:<br>Win64: 320 kB<br>Linux64: 1 MB |
| Customize Environment and Service Settings | Select **Yes**, to configure the settings defined in the domain source. |

**Table 4-1 Configuring the Domain Using the Configuration Wizard**

| Page in the Configuration Wizard Dialog Box | Action |
|---|---|
| Configure the Administration Server | Select or enter the administration server machine name or IP address for the Listen address.<br><br>**Note:** When you configure the administration server, we recommend that you accept the default server name (`AdminServer`). |
| Configure Managed Servers | Add as many managed servers as required.<br><br>**Note:** If you need a http router for load balancing, add it here. |
| Configure Clusters | Add a cluster and define the multicast address.<br><br>The multicast address is used by cluster members to communicate with each other. Clustered servers must share a single, exclusive multicast address. For each cluster on a network, the combination of multicast address and port must be unique. If two clusters on a network use the same multicast address, they should use different ports. If the clusters use different multicast addresses, they can use the same port or accept the default port, 7001. To support multicast messages, the administration server and the managed servers in a cluster must be located on the same subnet.<br><br>**Note:** WLI is intended to work with no more than one WLI cluster per domain. The domain can also include a WebLogic Server cluster. |
| Assign Servers to Clusters | Add all of the previously created managed servers to the cluster.<br><br>**Note:** If you had previously configured a managed server as an HTTP router, do not add it to the cluster. |
| Configure Machines | Configure the type of physical machines used in the cluster. |
| Assign Servers to Machines | Assign each instance of WebLogic Server to the machine in the cluster on which it runs.<br><br>**Note:** For information about creating a multi-node cluster (different managed servers on different machines) see "Creating and Starting a Managed Server on a Remote Machine: Main Steps" in *Creating Templates and Domains Using the pack and unpack Commands*. |
| Configure JDBC Data Sources | Configure each data source with information about the database that you want to use. |

**Table 4-1 Configuring the Domain Using the Configuration Wizard**

| Page in the Configuration Wizard Dialog Box | Action |
|---|---|
| Run Database Scripts | Run the scripts in the **cgDataSource-nonXA** and **p13nDataSource** data sources. |
| Configure JMS File Stores | Define configuration settings for the JMS file stores. |
| Review WebLogic Domain | Review the selections you made so far. |
| Create WebLogic Domain | On the Create WebLogic Domain page, specify the following values for each field and click **Create**:<br>• **Domain name:** Give a domain name<br>• **Domain Location:** `<Base Directory>\bea\user_projects\domains`<br>• **Application Location:** `<Base Directory>\bea\user_projects\applications`<br>After the domain is created successfully, select the **Start Admin Server** check box and then select **Done**. |

For information about configuring domains without using the Configuration Wizard, see "Tools for Configuring the Target Domain" in Creating and Configuring the WebLogic Domain in *Deploying WebLogic Platform Applications*.

## Editing Domain Configuration Files

Two configuration files are created in your WLI domain:

- `config.xml` contains a definition for the administration server and each managed server in the cluster, and it assigns the managed servers to the cluster.
- `wli-config.properties` contains other domain-specific information which WLI uses.

For information about specifying security features in your configuration by editing `config.xml`, see "Adding Proxy Server or Firewall Information to Domain Configuration." For more information about `config.xml`, see *WebLogic Server Configuration Reference*.

For information about `wli-config.properties`, see Appendix A, "wli-config.properties Configuration File."

### Adding Proxy Server or Firewall Information to Domain Configuration

If you will be using a Web service behind a proxy server or firewall, you must edit the `config.xml` file to include information about that proxy server or firewall.

To add proxy server or firewall information to your domain configuration, complete the following steps:

1. Open `config.xml` with an ASCII editor.
2. Find the line that starts with the following tag in the `config.xml` file:

   ```
   <Cluster
   ```

3. Add the following three attributes to the Cluster attribute list:

   ```
   FrontendHTTPPort="proxyPort" FrontendHTTPSPort="proxySSLPort"
   FrontendHost="proxyServerHost"
   ```

   For example, the following listing is an example of a cluster address with a firewall specified in a `config.xml` file for a cluster named `MyCluster` and a proxy server named `MyProxy`:

   ```
   <Cluster
   ClusterAddress="127.0.0.1:7001,127.0.0.2:7001,127.0.0.3,127.0.0.4:7001"
   FrontendHTTPPort="7006" FrontendHTTPSPort="7007" FrontendHost="MyProxy"
   MulticastAddress="127.0.0.5" MulticastPort="7010" Name="MyCluster"/>
   ```

4. Save your changes and close the `config.xml` file.

# Step 3. Configure WebLogic Integration Security

If you want to configure SSL for your cluster, you can do so by using the WebLogic Server Administration Console. For a domain in which security functionality is deployed in a multinode cluster, you also need to configure keystores, server certificate and private key for each managed server, and so on, for every machine in a cluster. You either need to use a separate keystore for each machine or you can use a single keystore if it is available to all machines.

The security administrator also has to make sure that the contents of shared or individual keystores in a cluster is consistent. Inconsistencies can be introduced when adding new certificates, if private keys must also be added. For example, if you add certificates for remote trading partners using the WebLogic Integration Administration Console, they can optionally be imported in the identity keystore used by each managed server in a cluster. However, this mechanism is not available (for security reasons) if private keys must be inserted in these keystores.

For information about the tasks you must complete, see:

- Configuring SSL in *Managing WebLogic Security.*
- Chapter 6, "Using WebLogic Integration Security."

For general information about configuring security for WebLogic Platform applications, see Configuring Security in *Deploying WebLogic Platform Applications*.

# Step 4. Start and Monitor the Managed Servers in the Domain

This section describes how to start the servers in your clustered domain:

- Starting the Managed Servers
- Monitoring and Shutting Down Your Servers

For information concerning starting servers for WebLogic Platform applications, see "Starting the Servers" in Creating and Configuring the WebLogic Domain in *Deploying WebLogic Platform Applications*.

## Starting the Managed Servers

To start servers in a domain for which the Node Manager is configured, complete the following procedure:

**Note:** You can also start managed servers from the command line by using the following command from `<domain-directory>/bin`:

```
startManagedWebLogic.cmd <managed-server-name> <admin-url>
```

1. If you have not done so already, start the Node Manager on each machine that hosts managed servers.

   For information about starting the Node Manager, see *Node Manager Administrator's Guide*.

2. If you have not done so already, start the WebLogic Server Administration Console.

   For the procedure to start the WebLogic Server Administration Console (and the administration server, if necessary), see "Starting the Administration Console" in Overview of WebLogic Server System Administration in *Configuring and Managing WebLogic Server.*

3. In the WebLogic Server Administration Console navigation tree, select the name of each managed server, in turn.
4. Select the **Configuration** tab, and then select the **Remote Start** tab. Set the properties for Node Manager to use for the managed server.

   For information about the setting the properties for Node Manager use, see Starting and Stopping Servers in *WebLogic Server Administration Console Online Help*.
5. Select the **Control** tab.
6. Click **Start this Server**.

   For information about how the Start Server command is affected by other settings made via the WebLogic Server Administration Console, see the WebLogic Server Administration Console Online Help.

# Monitoring and Shutting Down Your Servers

Once startup is complete, you can use the WebLogic Server Administration Console to verify deployments and status. For information about using WebLogic Server Administration Console to monitor your servers, see "Monitoring a WebLogic Server Domain" in WebLogic Server Performance and Tuning. For information about monitoring your WLI domain, see "Run-Time Tuning Issues" in Performance Tips.

**Note:** In cluster configurations, while running business processes or using the WebLogic Integration Administration Console, the following error message may appear in the WebLogic Server console window for the WebLogic Server that hosts the WebLogic Server Administration Console:

```
Failed to initialize clustered process configuration backend
```

If you encounter this problem, you must set the ClusterAddress attribute for the cluster. To learn how, see "Cluster Address" in Setting up WebLogic Clusters in *Using WebLogic Server Clusters*.

If you need to shut down your WLI application, use the WebLogic Server Administration Console.

**Note:** It is recommended that you do not close the command window or press Ctrl+c to stop WLI.

For the procedure to shut down your application gracefully, see "Graceful Shutdown of All Servers" and "Start/Stop a Server" in *WebLogic Server Administration Console Online Help*.

# Step 5. Deploy a WLI Application

Once you have configured and secured your WLI domain, you can deploy a WLI application to your cluster. You use the WebLogic Server Administration Console to deploy the EAR file that contains your WLI application.

The queues necessary for your application must be created manually using the WebLogic Server Administration Console.

**Note:** Async request and async request error queues, as well as conversational state tables, are created automatically for applications in the Workshop development environment. For production environments, however, you must create these queues and tables manually. For cluster deployments, these queues must be distributed destinations with physical members on each managed server.

For information about configuring these resources, see "Adding Resources Required by the Application From the wlw-manifext.xml File" in Creating and Configuring the WebLogic Domain in *Deploying WebLogic Platform Applications*.

For information about configuring JMS resources using the WebLogic Server Administration Console, see *Deploying Applications to WebLogic Server*.

**Note:** If your WLI solution uses the RDBMS Event Generator, be sure to configure the redelivery settings appropriately for its queues. For the procedure to configure the redelivery settings, see "RDBMS Event Generator" on page 3-10.

For the procedure to deploy an EAR file, see "Configuring and Deploying a New Enterprise Application or Web Service" in Enterprise Applications in *WebLogic Server Administration Console Online Help*.

For examples of automation scripts that build, configure, and deploy WLI applications outside of an interactive console environment, see the WLI Solution Samples and the PO Sample.

**Note:** Code samples and utilities are posted on dev2dev for your convenience. They are not products supported by BEA.

For a complete list of tools available to automate the application deployment process, see "Automating the Promotion Process" in Overview of WebLogic Platform Development in *Deploying WebLogic Platform Applications*.

CHAPTER 5

# Understanding WebLogic Integration High Availability

A clustered WebLogic Integration application provides scalability and high availability. A highly available deployment has recovery provisions in the event of hardware or network failures, and provides for the transfer of control to a backup component when a failure occurs.

For recommendations and database-specific requirements for configuring high availability WebLogic Integration applications, see Maintaining Availability.

## About WebLogic Integration High Availability

For a cluster to provide high availability, it must be able to recover from service failures. WebLogic Server supports failover for replicated HTTP session states, clustered objects, and services pinned to servers in a clustered environment.

For information about how WebLogic Server handles such failover scenarios, see Communications in a Cluster in *Using WebLogic Server Clusters*.

### Recommended Hardware and Software

The basic components of a highly available WebLogic Integration environment include the following:

- An administration server.
- A set of managed servers in a cluster.
- An HTTP load balancer (router).

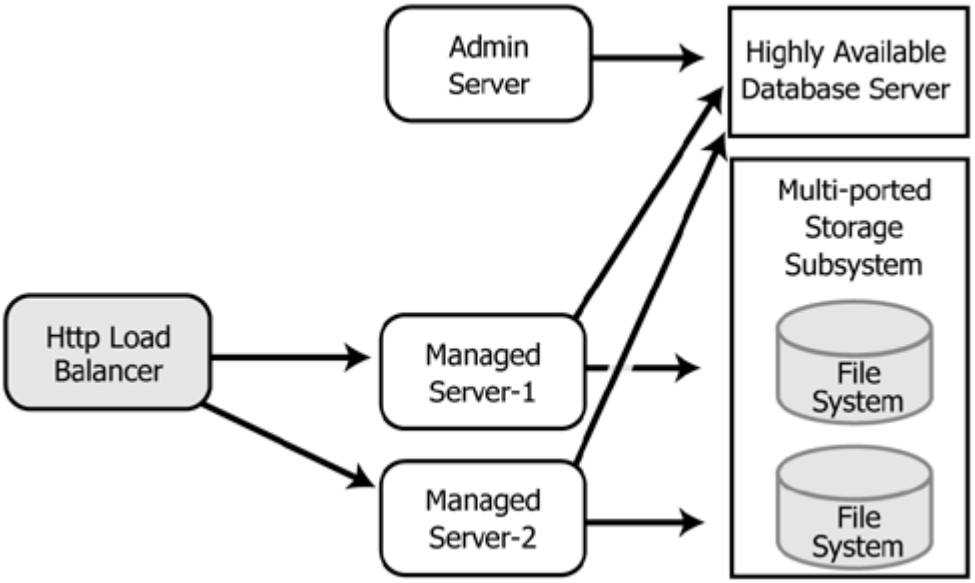
- Physically shared, highly-available disk subsystems for transaction recovery—Transaction logs from a failed server must be available to a managed server in order for migration to occur. A typical and recommended way to do this is by using a multi-ported disk subsystem or SAN, and allowing two or more servers to mount file systems within the disk subsystem. It is not necessary for the file system to be simultaneously shared; it is only necessary for one server to mount a file system at any one time.
- An IBM DB2, Microsoft SQL Server, Oracle, or Sybase database—You should take advantage of any high availability or failover solutions offered by your database vendor. (For database-specific information, see your database vendor's documentation.)

  **Note:** For information about availability and performance considerations associated with the various types of JDBC drivers, see in Configure JDBC Data sources in WebLogic Server Administration Console Online Help.

A full discussion of how to plan the network topology of your clustered system is beyond the scope of this section. For information about how to fully utilize load balancing and failover features for your Web application by organizing one or more WebLogic Server clusters in relation to load balancers, firewalls, and Web servers, see Cluster Architectures in *Using WebLogic Server Clusters*.

For a simplified view of a cluster, showing the http load balancer, highly available database and multi-ported file system, see the following figure.

**Figure 5-1 Simplified View of a Cluster**




## Regarding JMS File Stores

The default WebLogic Integration domain configuration uses a JDBC store for JMS servers. A file store can be used for JMS persistence in cases where a highly available multi-ported disk can be shared between managed servers, as described in the configuration shown in the preceding graphic. This will typically be more performant than a JDBC store.

For information about configuring JMS file stores, see *WebLogic Server Administration Console Online Help*.

# What Happens When a Server Fails

A server can fail due to either software or hardware problems. The following sections describe the processes that occur automatically in each case and the manual steps that must be taken in these situations.

## Software Faults

If a software fault occurs, the Node Manager (if configured to do so) will restart the WebLogic Server. For information about the Node Manager, see Overview of Node Manager. For

information about the steps to take to prepare for recovering a secure installation, see *Avoiding and Recovering From Server Failure*.

## Hardware Faults

If a hardware fault occurs, the physical machine may need to be repaired and could be out of operation for an extended period. In this case, the following events occur:

- The http load balancer will detect the failed server and will redirect to other managed servers. (The actual algorithm for doing this will depend on the vendor for the http load balancer.)
- All new internal requests, either RMI or JMS, will be redirected to other managed servers (JMS, if using distributed destinations).
- All in-flight transactions on the failed server are terminated.
- Another managed server can access process state, since it is held in the highly available database server. (A process will be in the state of the last successful transaction commit.)
- JMS messages that are already enqueued are not automatically migrated, but must be manually migrated. For more information, see "Server Migration" on page 5-4.

## Server Migration

In the case of a failure of extended duration, it may be necessary to migrate to another, operational managed server. When manually migrating a failed server to another managed server:

- The transaction logs from the failed server must be made available to the new migrated server. If you are using a shared disk subsystem, you should mount the file system from the failed server containing the transaction logs on the migrated server.
- The server must be manually migrated using the WebLogic Server Administration Console or, alternatively, through the command line utility.
- When JTA is migrated, it will read the TLOGs from the failed server and recover any in-doubt transactions. We recommend that you migrate JTA before migrating JMS.
- When JMS is migrated, it will allow access to the messages enqueued on the failed server.
- Any "singleton" Message Driven Beans (message driven beans that were tied to physical queues rather than distributed destination) will be activated, if the migrated JMS server contains the physical queues needed by the message driven beans.

For detailed information regarding WebLogic Server migration, see the following topics in the WebLogic Server documentation set:

- Failover and Replication in a Cluster in *Using WebLogic Server Clusters*
- *Avoiding and Recovering From Server Failure*

# Failure and Recovery: Trading Partner Integration

In addition to the high availability features of WebLogic Server, WebLogic Integration has failure and recovery characteristics that are based on the implementation and configuration of your WebLogic Integration solution.

For more information about WebLogic Integration failure and recovery topics, see WebLogic Integration Application Recovery in the *WebLogic Integration Solutions Best Practices FAQ*.

RosettaNet and ebXML handle failure and recovery differently because of differences in the business protocols. However, both protocols send messages that fail to be delivered after the configured number of retries to `wli.b2b.failedmessage.queue`. If you require additional processing of failed messages, you can implement custom message listeners for this queue.

## RosettaNet

When message delivery fails in the case of RosettaNet messages, the WebLogic Integration protocol layer does not retry messages. It returns HttpStatus code to the workflow layer, instead. RosettaNet workflows are usually designed to handle retries.

The WebLogic Integration Administration Console enables you to specify retry intervals, retry counts, and process timeouts for various trading partners based on the PIP(s) being used. For example, RosettaNet typically supports three retries at two-hour intervals with an overall 24-hour limit on the life of the actual PIP exchange. For information about changing these settings, see "Viewing and Changing Bindings" in Trading Partner Management in *Using the WebLogic Integration Administration Console*.

If one instance of WebLogic Integration sends a message to another instance and the destination instance has failed, you may see one or more error messages followed by a stack trace in the server console.

## ebXML

You can specify ebXML message retries using the WebLogic Integration Administration Console, Trading Partner Management Bulk Loader, or third-party Trading Partner Management

message beans. If you set ebXML Delivery Semantics to `OnceAndOnlyOnce` or `AtLeastOnce`, messages will be retried according to the values you specify for Retry Count and Retry Interval. For information about using the WebLogic Integration Administration Console to set ebXML message retries, see "Defining Protocol Bindings" in Trading Partner Management in *Using the WebLogic Integration Administration Console*.

For ebXML processes, set the action mode value to non-default to guarantee recovery and high availability. For information about setting the action mode, see "ebXML Business Processes" in Introducing ebXML Solutions in *Introducing Trading Partner Integration*.

CHAPTER 6

# Using WebLogic Integration Security

The following sections describe how to set up and manage security for WebLogic Integration solution deployments:

- Overview of WebLogic Integration Security
- Considerations for Configuring Security
- Setting Up a Secure Deployment

Before you proceed with the remainder of this topic, be sure to read Securing WebLogic Server.

## Overview of WebLogic Integration Security

The foundation of every secure deployment of a WebLogic Integration solution is the set of security features provided by WebLogic Server. After you configure security for the underlying WebLogic Server layer of your environment, you need to configure and manage security for those WebLogic Server entities that are specific to WebLogic Integration:

- Users of the WebLogic Integration solution and WebLogic Integration Administration Console, the groups to which they belong, and the roles they are assigned.
- Trading partners for which security management is particularly important, because trading partners are required to produce digital certificates for sending and receiving business messages in a secure environment.

As the security administrator for your environment, you need to focus your efforts on a set of predefined principals and resources that are created along with a WebLogic Integration domain.

This introduction presents the following topics to give you a high-level view of WebLogic Integration security:

- Security and WebLogic Integration Domains
- WebLogic Server Security Principals and Resources Used in WLI

**Note:** For a secure deployment, avoid running WebLogic Integration in the same WebLogic Server instance as any applications for which security is not provided. Internal WebLogic Integration API calls are not protected from such collocated applications.

# Security and WebLogic Integration Domains

When you create a WebLogic Integration domain using the Domain Configuration Wizard, the domain is configured to include:

- Default WebLogic Integration roles and groups. Default security policies define the roles authorized to access specific WebLogic Integration resources. For more information, see User Management in *Managing WebLogic Integration Solutions*.
- PasswordStore, which is described in "WebLogic Integration PasswordStore for Encrypted Passwords".
- Default identity and trust keystores, which are described in "Keystore for Private Keys and Certificates".
- B2BDefaultWebAppApplication, on which you can configure policies to authorize access to Trading Partner Integration.

For information about using the Configuration Wizard, see *Creating WebLogic Domains Using the Configuration Wizard*.

## WebLogic Integration PasswordStore for Encrypted Passwords

All passwords are kept in encrypted form in the PasswordStore. WebLogic Integration does not require clear-text passwords. The PasswordStore the uses Sun JCE provider for password-based encryption. Access to passwords is controlled through an MBean API and passwords are accessed using password-aliases.

You use the WebLogic Integration Administration Console to manage passwords in the PasswordStore. For more information, see the following topics in Trading Partner Management in *Using the WebLogic Integration Administration Console*:

- Adding Passwords to the PasswordStore

- Listing and Locating Password Aliases
- Changing the Password for a Password Alias
- Deleting Passwords from the PasswordStore

## Keystore for Private Keys and Certificates

WebLogic Integration requires that you use keystores to store all private keys and certificates. A keystore is a protected database that holds keys and certificates. If you have keys and certificates and use message encryption, digital signatures, or SSL, you must use a keystore for storing those keys and certificates and make the keystore available to applications that might need it for authentication or signing purposes.

### Types of Keystores

When you set up a WebLogic Integration domain for trading partner integration collaborations, the following keystores are configured.

**Table 6-1 Types of Certificates Used in WebLogic Integration**

| Type of KeyStore | Description |
|---|---|
| Identity keystore | Stores private keys for local trading partners and certificates for both the local trading partner and remote trading partners. Certificates are of the following types:<br>• client<br>• server<br>• signature<br>• encryption<br>WebLogic Integration retrieves private keys and certificates from this keystore to use for SSL, message encryption, and digital signatures. For more information about certificates, see "About Digital Certificates" on page 6-6. |
| Trust keystore | WebLogic Server uses the trust keystore to locate trusted CAs for SSL. WebLogic Integration uses it to locate the trusted CAs while verifying signature and encryption. |

### Default Keystores for the Test Environment

When you create a new domain using the WebLogic Platform Configuration Wizard and the WebLogic Integration template, the new domain contains Demo Keystores of type JKS. The Demo KeyStores performs the following actions:

- Utilizes the JDK bundled Java KeyStore (JKS) provider, which implements the keystore as a file.
- Protects each private key with an individual password.
- Protects the entire keystore with a password.

### Keystores in a Production Environment

You can use the Demo keystores in a development or testing environment, but you must either create or use existing identity and trust keystores suitable for production environment. To create a keystore and make it available for trading partner integration:

1. If the identity and trust keystores do not already exist in your domain, create them according to the instructions in "Storing Private Keys, Digital Certificates, and Trusted Certificate Authorities" in Configuring SSL in *Securing WebLogic Server* .
2. Configure the keystores using the WebLogic Server Administration Console according to the instructions in "Configuring KeyStores" in Configuring SSL in *Securing WebLogic Server* .
3. Add trading partner certificates to the identity keystore. For more information, see "Step 3: Configure Web Application and Web Service Security-Related Deployment Descriptors" on page 6-11.
4. Add trusted certificate authority certificates to the trust keystore.

For information about refreshing the keystore using the WebLogic Integration Administration Console, see "Refreshing the Keystore" in Trading Partner Management in *Using the WebLogic Integration Administration Console*.

In a clustered domain, you need to create and configure a separate keystore for each WebLogic Server.

## WebLogic Server Security Principals and Resources Used in WLI

WebLogic Integration supports role-based authorization. Although the specific users (*principals*) that require access to the components that make up your WebLogic Integration application may

change depending on the deployment environment, the roles that require access are typically more stable. Authorization involves granting an entity permissions and rights to perform certain actions on a resource.

In role-based authorization, security policies define the roles that are authorized to access the resource. In addition to the built-in roles that are associated with certain administrative and monitoring privileges, security policies that control access to the following resources can be configured from the WebLogic Integration Administration Console:

- *Process operations*

  Policies define the role required to invoke the process operations. For more information on the policies you can set, see "Process Security Policies" under "About Process Configuration" in Processes Configuration in *Using the WebLogic Integration Administration Console*.

- *Message Broker channels*

  Policies define the roles required to subscribe and publish to a given channel. For more information, see "About Message Broker Channels" in Message Broker in *Using the WebLogic Integration Administration Console*.

- *Trading Partner Integration Transport*

  Policies define the roles required to accept messages from remote trading partners at URIs in `B2BDefaultWebAppApplication`.

Once the roles required for access are set, the administrator can map users or groups to the roles as required.

Unlike membership in a group, which is directly assigned, membership in a security role is dynamically calculated based on the set of conditions that define the role statement. Each condition specifies user names, group names, or time of day. When a principal (user) is "in" a role based on the evaluation of the role statement, the access permissions of the role are conferred on the principal.

# Considerations for Configuring Security

Before you configure the security for your WebLogic Integration domain, consider the following:

- About Digital Certificates
- Using the Secure Sockets Layer (SSL) Protocol
- Using an Outbound Proxy Server or Proxy Plug-In

- Using a Firewall

The following sections present a high-level discussion of these considerations and describe how they affect your WebLogic Integration security configuration.

# About Digital Certificates

Digital certificates are electronic documents used to identify principals and objects as unique entities over networks such as the internet. A digital certificate securely binds the identity of a user or object, as verified by a trusted third party known as a certificate authority, to a particular public key. The combination of the public key and the private key provides a unique identity for the owner of the digital certificate.

When you set up a WebLogic Integration environment as the foundation of your inter-enterprise commerce, using Trading Partner Integration capabilities, you need to obtain and configure a specific set of digital certificates and keys. This set includes the following:

- Server certificate—Required for SSL for the WebLogic Server instance on the local machine.
- Root Certificate Authority—Trusted third-party organization or company that is willing to vouch for the identities of those to whom it issues digital certificates and public keys. Verisign and Baltimore are examples of CAs.
- Trading partner certificates—Required for each local and remote trading partner that is involved in Trading Partner Integration collaborations. These certificates include the client certificate; they may also include encryption and signature certificates, as well. They are used for authentication, authorization, signature support, and message encryption.

## Digital Certificate Formats

Make sure that the formats and packaging standards of your digital certificates are compatible with WebLogic Server. Digital certificates have various encoding schemes, including the following:

- Privacy Enhanced Mail (PEM)
- Definite Encoding Rules (DER)
- Public Key Cryptography Standards 7 and 12 (PKCS7 and PKCS12)

The public key infrastructure (PKI) in WebLogic Server recognizes digital certificates that comply with either versions 1 and 3 of X.509, X.509v1 and X.509v3. We recommend obtaining digital certificates from a certificate authority, such as Verisign or Entrust.

**Note:** If a trading partner in a conversation uses Microsoft IIS as a proxy server, all the certificates used in the conversation must be trusted by a well-known Certificate Authority, such as Verisign or Entrust. The use of self-signed certificates will cause a request passed through the IIS proxy server to fail. This is a restriction in IIS, not WebLogic Integration.

For more details, see "Transport-Level Security" in Trading Partner Integration Security in *Introducing Trading Partner Integration*.

# Using the Secure Sockets Layer (SSL) Protocol

The SSL protocol provides secure connections by supporting two functions:

- It enables each of two applications linked through a network connection to authenticate the other's identity
- It encrypts the data exchanged between the applications for each trading partner using SSL.

An SSL connection begins with a handshake during which the applications exchange digital certificates, agree on the encryption algorithms to be used, and generate encryption keys that are then used for the remainder of the session.

If you are using SSL for trading partner authentication and authorization, which we strongly recommend for Trading Partner Integration collaborations, you need to configure the following:

- SSL for each machine in your WebLogic Integration domain.
- Set of digital certificates and private keys for each trading partner
- Server certificate for each machine in the WebLogic Integration domain
- Certificates of trusted Certificate Authorities (CA)

Not required by SSL, but strongly recommended, is the creation and use of identity and trust keystores for storing all the certificates and keys used in your WebLogic Integration domain. For more information about SSL, certificates, and keystores, see Configuring SSL in *Securing WebLogic Server* .

# Using an Outbound Proxy Server or Proxy Plug-In

This section discusses the implications of using either an outbound proxy server or the WebLogic proxy plug-in.

## Using an Outbound Proxy Server

A proxy server allows trading partners to communicate across intranets or the Internet without compromising security. If you are using WebLogic Integration in a security-sensitive environment, you may want to use WebLogic Integration behind a proxy server. Specifically, a proxy server is used to:

- Hide, from external hackers, the local network addresses of the WebLogic Server instances that host WebLogic Integration
- Restrict access to the external network
- Monitor external network access to the local instances of WebLogic Server that host WebLogic Integration

When proxy servers are configured on the local network, network traffic (sent with the SSL and HTTP protocols) is tunneled through the proxy server to the external network.

If an outbound proxy server is used in your environment, be careful when specifying the transport URI endpoints for the local trading partner. If you are using an HTTPS proxy, then you need to specify the `ssl.ProxyHost` and `ssl.ProxyPort` Java system properties. For details, see "Configuring Trading Partner Integration to Use an Outbound HTTP Proxy Server" in Trading Partner Integration Security in *Introducing Trading Partner Integration*.

## Using a Web Server with the WebLogic Proxy Plug-In

As an alternative to using an outbound proxy server, you may want to configure WebLogic Integration with a Web server, such as an Apache server, that is programmed to handle business messages from a remote trading partner. The Web server can provide the following services:

- Receive business messages from a remote trading partner
- Authenticate digital certificates from the trading partner

The Web server then uses the WebLogic proxy plug-in, which you can configure to provide the following services:

- Forwarding of business messages received by the Web server to WebLogic Integration, which is running inside a secure internal network.
- Extraction of the remote trading partner certificate from the Web server and delivery of it to WebLogic Server for authentication. WebLogic Integration can then authenticate the trading partner certificate and business message.

To configure the WebLogic proxy plug-in, perform the following actions:

- Make sure you configure the proxy server with the WebLogic proxy plug-in to direct requests to WebLogic Server.
- Decide which protocol you want to use for the network connection between the proxy server and the WebLogic Integration domain. The default protocol is HTTP. Configure the proxy plug-in to use one-way SSL only if you prefer to use SSL.
- When configuring the transport for remote trading partners, specify the remote URI endpoint with the HTTPS protocol, even though the HTTP protocol is used in the network connection between the WebLogic proxy plug-in and the WebLogic Integration domain.
- When relaying a business message from one trading partner to another, some proxy servers include only the leaf certificate, instead of the entire CA certificate chain. In such instances, trading partner authentication may fail. To prevent such failures, we recommend you specify the leaf certificate as the trusted CA certificate. (For more information about leaf certificates, see "Certificate Authorities" in Trading Partner Integration Security in *Introducing Trading Partner Integration*.)
- If the local trading partner site uses a Web server configured with a WebLogic proxy plug-in, specify the trading partner transport URI endpoints in the usual manner.
- If the remote trading partner is also using WebLogic Integration, but is using a proxy server other than the WebLogic proxy server, then it is likely that the remote site is configured with the WebLogic proxy plug-in. When you are configuring a remote trading partner under these circumstances, you must specify the host and port of the trading partner's proxy server as the transport URI endpoints. The WebLogic proxy plug-in performs the necessary URL transformations to business messages received for that remote trading partner.

## Using a Firewall

If your WebLogic Integration environment is configured with a firewall, make sure your firewall is configured properly so that business messages can flow freely to and from local trading partners via the HTTP or HTTPS protocols.

# Setting Up a Secure Deployment

The following sections provide instructions for the tasks you must complete to set up a secure deployment:

- "Step 1: Create the Domain"
- "Step 2: Configure WebLogic Server Security"

- "Step 3: Configure Web Application and Web Service Security-Related Deployment Descriptors"
- "Step 4: Configure Security Policies and Manage Users"
- "Step 5: Configure Worklist Security"
- "Step 6: Configure Trading Partner Integration Security"

# Step 1: Create the Domain

Create the WebLogic Integration domain using the Domain Configuration Wizard, as described in Chapter 2, "Configuring a Single-Server Deployment" or Chapter 4, "Configuring a Clustered Deployment".

**Note:** We recommend that you configure your domain with SSL enabled.

The WebLogic Server Administration Console enables you to make additional customizations to your WebLogic Integration domain and default security realm.

For information about customizing security features using the WebLogic Server Administration Console, see "Customizing the Default Security Configuration" in *Managing WebLogic Security*.

For a description of how to add firewall information to your domain configuration file, see "Adding Proxy Server or Firewall Information to Domain Configuration".

# Step 2: Configure WebLogic Server Security

When configuring WebLogic Server security, be sure to do the following:

1. Obtain the server certificates for the local and remote trading partners. For SSL, server certificates are required for each instance of WebLogic Server involved in a trading partner request.
2. Consider the following questions and take the appropriate actions for your environment:
   - Does the common name of the certificate match the host name of the machine on which the corresponding instance of WebLogic Server is running?

     If the two names are not the same, then the local WebLogic Server instance must be configured with hostname verification disabled. This requirement applies to the server certificate for *any* trading partner in any Trading Partner Integration. You can disable hostname verification in the WebLogic Server Administration Console by checking the Hostname Verification Ignored attribute on the SSL tab for the Server node.

**Note:** We do not recommend configurations that require you to disable hostname verification. Hostname verification prevents some types of security attacks.

– Are the formats of the server certificate and private key for a remote trading partner supported by WebLogic Server?

"About Digital Certificates" on page 6-6 lists the supported certificate formats. For server certificates, PEM encoded X.509 V1 or V3 is the most commonly accepted format by SSL servers.

For private keys, PKCS8, which is the password-encrypted private key, is the most common format. Be sure to set the private key password so that WebLogic Server can read the private key.

– What is the CA certificate chain for the WebLogic Server server certificate?

A certificate chain is an array of digital certificates for trusted CAs, each of which is the issuer of the previous digital certificate.

You may specify one file containing all the intermediate and root CA certificates. (Note that if the file contains more than one CA certificate, WebLogic Server requires a PEM encoded file.) If you use the trust keystore to store trusted CA certificates, be sure to import the whole chain in to the trust keystore.

3. Configure the WebLogic identity and trust keystores. For information on creating and configuring keystores, see Configuring SSL in *Securing WebLogic Server*.

   **Note:** Note the following considerations for using keystores:

   - One caveat to using a keystore is that once you import a key and certificate with an alias into a keystore, overwriting that certificate file with a new certificate does not import the new certificate into the keystore.
   - Make sure that your keystore is up-to-date with your current set of certificates and keys, and make sure that the WebLogic Integration repository reflects the relevant content of your keystore.

# Step 3: Configure Web Application and Web Service Security-Related Deployment Descriptors

Using Workshop, a developer can edit web application settings and wb service security-related deployment descriptors in the following three XML files before building and packaging the EAR file that contains your WebLogic Integration application:

- `web.xml`

- `weblogic.xml`
- `wlw-config.xml`

A system administrator at deployment time may have more information regarding the production environment and security requirements. Under these circumstances, you can reconfigure the Web application settings and Web service security-related deployment descriptors in your EAR file as necessary by performing the following procedure.

**Note:** A developer typically adds any Service Broker control, Process control or callback selector annotations that are necessary for security to `jcx` or `jpd` files before packaging the EAR for deployment. However, these annotations can also be added or changed during this procedure.

1. Explode the EAR file that contains your WebLogic Integration application, and verify the configuration of the following items:

   In `web.xml` and `weblogic.xml`, security-related deployment descriptors for Web applications that contain JPDs should be set to appropriate user credentials, method of authentication, and location of resources.

   For information about these deployment descriptors, see "Web Application Security-Related Deployment Descriptors" in Securing Web Applications in *Programming WebLogic Security.*

2. Repackage the EAR, and deploy it to the production WebLogic Integration domain using the WebLogic Server Administration Console.

**WARNING:** Redeploying an application from Workshop causes a loss of security authorizations. Deploy the EAR file using the WebLogic Server Administration Console to preserve your security authorizations.

For information about packaging and deploying EAR files, see Running and Debugging Applications.

# Step 4: Configure Security Policies and Manage Users

Once the WebLogic Integration application has been deployed on your production hardware, you can use the WebLogic Integration Administration Console to configure security policies and manage users.

For the procedure to start the WebLogic Server Administration Console see Start the Console in *Managing WebLogic Integration Solutions*.

The following sections provide instructions for the tasks you must complete to configure security policies and manage users:



## Configuring Security Policies for Business Processes

1. On the home page of the WebLogic Integration Administration Console, click **Process Configuration**.

   The **Process Property Summary** page lists every business process in the WebLogic Integration application.

2. Click the display name of a process to access the **Process Type Details** page.

   From the **Process Type Details** page, you can configure the following security settings for the business process:

   – Dynamic Client Callback Properties

   – Execution Policy

   – Process Authorization Policy

   – Method Authorization Policy

   – Control Callback Authorization Policy

   For descriptions of these settings and the procedures to configure them, see "Viewing and Changing Process Details" in Processes Configuration in *Using The WebLogic Integration Administration Console*.

3. Click **View Process Summary** to return to the **Process Property Summary** page, and repeat step 2 for each business process.

4. After configuring each business process, click **View Dynamic Controls**.

   The **View Dynamic Controls Summary** page lists every dynamic control in the WebLogic Integration application.

5. Select the **Edit** link to the right of a selector value to display the **Edit Service Broker Control Selector** or **Edit Process Control Selector** page, depending on the type of the control.

On these pages, you can configure the client certificate or username/password settings used in outbound calls by the selected service broker or process control. For descriptions of these settings and the procedures to configure them, see "Adding or Changing Dynamic Control Selectors" in Processes Configuration in *Using The WebLogic Integration Administration Console*.

6. Click **View Dynamic Controls** to return to the **View Dynamic Controls Summary** page, and repeat step 5 for each dynamic control.
7. After configuring the security settings for each business process and dynamic control, click **Home** in the module navigation bar to return to the home page.

## Configuring Security Policies for Message Broker Channels

1. On the home page of the WebLogic Integration Administration Console, click **Message Broker**.

   The **Channel Summary List** page displays every channel in the Message Broker.
2. Click a channel name to access the **View Channel Details** page, and then click **Edit Security Details**.

   On the **Edit Channel Subscribe and Publish Policies** page, you can configure the following security settings for the channel:

   – Publish Roles

   – Subscribe Roles

   – Dispatch As (the user under which messages are sent to subscribers)

   For descriptions of these settings and the procedures to configure them, see "Viewing and Changing Process Details" in Processes Configuration in *Using THe WebLogic Integration Administration Console*.
3. Click **View All** to return to the **Channel Summary List** page, and repeat step 2 for each channel.
4. After configuring the security settings for each channel, click **Home** in the module navigation bar to return to the home page.

## Managing Production Users

1. On the home page of the WebLogic Integration Administration Console, click **User Management**.

The **View and Edit Users** page displays a list of all users within WebLogic Integration. From this page you can create new users, delete users, or access details—including group membership—for a selected user.

For information about managing users, see the following topics in User Management in *Managing WebLogic Integration Solutions*:

- Adding a User
- Viewing and Changing User Properties

2. Choose the **Groups** tab.

   The **View and Edit Groups** page displays a list of all groups within WebLogic Integration. From this page you can create new groups, delete groups, or access details—including group membership—for a selected group.

   For information about managing groups, see the following topics in User Management in *Managing WebLogic Integration Solutions*:

   - Adding a Group
   - Viewing and Changing Group Properties

3. Choose the **Roles** tab.

   The **View and Edit Roles** page displays a list of all roles within WebLogic Integration. From this page you can create new roles, delete roles, or access details—including role conditions—for a selected role.

   For information about managing roles, see the following topics in User Management in *Managing WebLogic Integration Solutions*:

   - Adding a Role
   - Viewing and Setting Role Conditions

# Step 5: Configure Worklist Security

WebLogic Integration domains includes the following default WebLogic Integration groups and roles that have access to worklist functionality:

- Admin— This is defined by WebLogic Server, this role is listed in the default administrative policy for all Worklist resources.
- IntegrationAdmin— This is defined by WebLogic Server, this role is listed in the default administrative policy for all Worklist resources.

- IntegrationUser— This is defined by WebLogic Server, this role is listed in the default query policy for TaskPlan resources.
- TaskCreationRole—Defined by WLI, this role is listed in the default task creation policy for all task plans.

The process of configuring worklist security is basically one of assigning users to groups, groups to roles, and ensuring that those roles have appropriate permission levels by defining policies. (For information on how to make these assignments, see "Managing Production Users" on page 6-14.) Once you have configured worklist security, you can manage owners for tasks in a worklist.

The WebLogic Integration Administration Console provides tools that allow you to manage users, groups, roles, and policies, along with worklist task ownership.

## Step 6: Configure Trading Partner Integration Security

WebLogic Integration solutions that involve the exchange of messages between trading partners across firewalls have special security requirements, including trading partner authentication and authorization, as well as nonrepudiation.

To configure Trading Partner Integration security, you must perform the following tasks:

- Obtain the certificates and keys required for conducting Trading Partner Integration collaborations. Required certificates include those for the trusted CAs, as well as the trading partner certificates and keys mentioned earlier, and the server certificate and key for each instance of WebLogic Server used in your environment.
- Configure keystores to store certificates and private keys used in a WebLogic Integration environment.
- Configure local trading partners.
- Configure remote trading partners.
- Configure security for the business protocols used including transport level security and message level security.
- Implement the security requirements for the business protocols used.

For detailed information and procedures regarding configuration of Trading Partner Integration, see Trading Partner Integration Security in *Introducing Trading Partner Integration*.

APPENDIX A

# wli-config.properties Configuration File

The `wli-config.properties` file controls various run-time characteristics of WebLogic Integration. The file is read when WebLogic Server starts. WebLogic Integration does not implement changes made to the file while WebLogic Server is running.

A related file, `jws-config.properties`, specifies domain-wide configuration parameters for the Workshop run-time engine.

The following table describes the contents and default settings of `wli-config.properties`.

**Table C-2 Contents of wli-config.properties file**

| Property | Description | Default value |
|---|---|---|
| `weblogic.wli.DocumentMaxInline Size` | Minimum size in bytes for documents stored in the SQL Document Store | `524288` |
| `weblogic.wli.DocumentMaxIn MemorySize` | Maximum size in bytes of document buffered before writing to the SQL Document Store | `524288` |
| `weblogic.wli.ProcessDocument CleanupIntervalMinutes` | Interval in minutes at which unreferenced documents are deleted from the SQL Document Store | `20` |
| `weblogic.wli.SQLStoreLogSQ` | Request that the SQL Document Store log its SQL requests to the Workshop debug log | `false` |

**Table C-2 Contents of wli-config.properties file**

| Property | Description | Default value |
|---|---|---|
| `weblogic.wli.OracleLog` | Request an `oracle.log` in the domain directory | `false` |
| `weblogic.wli.ProcessTypeMaxNameLength` | Maximum size of process (workflow) type names, reflected in WLI system tables.<br>**Note:** Delete the default value and then leave the value for this property unspecified to get a value appropriate for the type of database in use. | `200` |
| `weblogic.wli.PurgeRowNumMax` | Maximum number of rows in the result set that returns instances ready to be purged | `20` |
| `weblogic.wli.TrackingPurgeTxn TimeMillis` | Approximate transaction duration in milliseconds for purging tracking data | `250` |
| `weblogic.wli.WliClusterName` | In a domain with multiple clusters, this property represents the name of the cluster running the single WLI instance.<br>**Note:** Specify a cluster name only if running in a domain with multiple clusters. | None provided |
| `wli.jmseg.EatSoapActionElement` | When set to `true`, an event generator will consume the first element under the `<SOAP:Body>` element. This makes the event generator consistent with the JWS/JPD SOAP protocol, which uses the first element under the `<SOAP:Body>` element to bind to a method. | `true` |

APPENDIX B

# WebLogic Integration Deployment Resources

This section describes the WebLogic Integration deployment resources, including targeting information for deployment in a cluster configuration. The table below contains the following information about each resource:

- Resource Target—The target for a resource depends on the resource's type.
  - For queues and topics, the target is a single, migratable JMS server, a JMS server on the administration server, or a distributed destination.

    For information about migratable JMS servers and distributed destinations, see *Programming WebLogic JMS*.
  - For all other resources (including deployments, JMS and JDBC services) the target is either the administration server or a cluster.
- Resource—The name of the resource as shown in the WebLogic Server Administration Console and the individual package or service (in a resource group).

  Note that some resources contain abbreviations that are a legacy from prior WebLogic Integration releases: `b2b` corresponds to Trading Partner Integration.
- Administration Console Navigation—Route through the WebLogic Server Administration Console navigation tree to the specified package or service. All resources can be viewed and modified in the WebLogic Server Administration Console.

**Table D-1 WebLogic Integration Deployment Resources**

| Resource Target | Resource | Description | Administration Console Navigation |
|---|---|---|---|
| Admin Server (Applications) | `calendar-ejbs-generic.ear` | WLI Calendar Persistence | Domain > Deployments |
| | `applications/<domain>/B2BDefaultWebApp` | B2B Default Web Application | Domain > Deployments |
| | `wliconsole.war` | WLI Console | Domain > Deployments |
| | `tibRVEG_ear` | TIBCO EG Application | Domain > Deployments |
| | `mqEG_ear` | MQ EG Application | Domain > Deployments |
| | `httpEG_ear` | `HTTP EG Application` | Domain > Deployments |
| | `rdbmsEG_ear` | RDBMS EG Application | Domain > Deployments |
| | `console/worklistconsoleEar` | Worklist Console | Domain > Deployments |
| | `worklist-admin.ear` | Worklist Admin Application | Domain > Deployments |
| Admin Server (Resources) | `wli.internal.b2b.events.topic` | Topic for cluster events | Domain > Services > Messaging > JMS Modules > conversational-jms |
| | `wli.internal.configfile.request.queue` | Return queue to configuration manager | Domain > Services > Messaging > JMS Modules > conversational-jms |
| | `wli.internal.configfile.update.topic` | Update topic for configuration manager | Domain > Services > Messaging > JMS Modules > conversational-jms |

**Table D-1 WebLogic Integration Deployment Resources (Continued)**

| Resource Target | Resource | Description | Administration Console Navigation |
|---|---|---|---|
| | cgQueue | WLI/WLW Connection factory | Domain > Services > Messaging > JMS Modules > conversational-jms |
| | com.bea.wli.b2b.serve.TopicConnectionFactory | WLI-B2B System Topic Factory | Domain > Services > Messaging > JMS Modules > conversational-jms |
| | bpmArchDataSource | WLI JDBC Data source for Archiving Database Tables | Domain > Services > JDBC > Data Sources |
| | cgDataSource | WLI/WLW JDBC Data source | Domain > Services > JDBC > Data Sources |
| | cgDataSource-nonXA | WLI/WLW JDBC Data source | Domain > Services > JDBC > Data Sources |
| | p13nDataSource | P13N Data source used by worklist | Domain > Services > JDBC > Data Sources |
| | com.bea.wli.init.BPMStartupAfterActivation | WLI Post-Activation Startup Class | Domain > Environment > Startup & Shutdown Classes |
| | com.bea.wli.init.BPMShutdown | WLI Shutdown Class | Domain > Environment > Startup & Shutdown Classes |
| | com.bea.wli.init.BPMStartup | WLI Startup Class | Domain > Environment > Startup & Shutdown Classes |
| | All Libraries | The system libraries | Domain > Deployments |

**Table D-1 WebLogic Integration Deployment Resources (Continued)**

| Resource Target | Resource | Description | Administration Console Navigation |
|---|---|---|---|
| Cluster (Applications) | `jpd-ejbs.ear/wliadmin` | WLI Admin | Domain > Deployments > WLI System EJBs |
| | `jpd-ejbs.ear/adminhelper` | WLI Admin Helper | Domain > Deployments > WLI System EJBs |
| | `jpd-ejbs.ear/calendar/generic` | WLI Calendar Persistence | Domain > Deployments > WLI System EJBs |
| | `jpd-ejbs.ear/ebxml` | WLI ebXML | Domain > Deployments > WLI System EJBs |
| | `jpd-ejbs.ear/message-tracking` | WLI Message Tracking | Domain > Deployments > WLI System EJBs |
| | `jpd-ejbs.ear/proxydispatcher` | WLI Process Proxy Dispatcher | Domain > Deployments > WLI System EJBs |
| | `jpd-ejbs.ear/tracking` | WLI Process Tracking | Domain > Deployments > WLI System EJBs |
| | `jpd-ejbs.ear/rosettanet` | WLI RosettaNet | Domain > Deployments > WLI System EJBs |
| | `jpd-ejbs.ear/sync2AsyncIM` | SyncAsyncTransportServlet | Domain > Deployments > WLI System EJBs |
| | `jpd-ejbs.ear/SyncAsyncTransportServlet` | SyncAsyncTransportServlet | Domain > Deployments > WLI System EJBs |
| | `jpd-ejbs.ear/transport/responsehandler` | SyncAsync Response Handler EJB | Domain > Deployments > WLI System EJBs |
| | `jpd-ejbs.ear/b2btransport-webapp` | WLI-B2B HTTP Transport | Domain > Deployments > WLI System EJBs |
| | `calendar-ejbs-generic.ear` | WLI Calendar Persistence | Domain > Deployments |

**Table D-1 WebLogic Integration Deployment Resources (Continued)**

| Resource Target | Resource | Description | Administration Console Navigation |
|---|---|---|---|
| | `worklist-ejbs-worksub.ear` | Worklist Work Substitute EJB | Domain > Deployments |
| | applications/<domain>/B2BDefaultWebApp | B2B Default Web Application | Domain > Deployments |
| | `QueueTransportEJB.jar` | JWSQueueTransport | Domain > Deployments |
| Cluster (Resouurces) | `com.bea.wli.init.BPMStartupAfterActivation` | WLI Post-Activation Startup Class | Domain > Environment > Startup & Shutdown Classes |
| | `com.bea.wli.init.BPMShutdown` | WLI Shutdown Class | Domain > Environment > Startup & Shutdown Classes |
| | `com.bea.wli.init.BPMStartup` | WLI Startup Class | Domain > Environment > Startup & Shutdown Classes |
| | `cgQueue` | WLI/WLW Connection factory | Domain > Services > Messaging > JMS Modules > conversational-jms |
| | `wli.internal.egrdbms.XAQueueConnectionFactory` | RDBMS Event Generator Connection Factory | Domain > Services > Messaging > JMS Modules > conversational-jms |
| | `bpmArchDataSource` | WLI Data Source for Archiving | Domain > Services > Messaging > JMS Modules > conversational-jms |
| | `cgDataSource` | WLI/WLW Data Source | Domain > Services > JDBC > Data Sources |
| | All Libraries | The system libraries | Domain > Deployments |

**Table D-1 WebLogic Integration Deployment Resources (Continued)**

| Resource Target | Resource | Description | Administration Console Navigation |
|---|---|---|---|
| | `wli.internal.msgtracking.queue` | Message tracking JMS queue | Domain > Services > Messaging > JMS Modules > conversational-jms |
| | `wli.b2b.mt.event.stream` | Business process message tracking event queue | Domain > Services > Messaging > JMS Modules > conversational-jms |
| | `wli.b2b.mt.event.stream_error` | Errors associated with business process message tracking event queue | Domain > Services > Messaging > JMS Modules > conversational-jms |
| | `wli.internal.b2b.ebxmlencoder.queue` | ebXML outbound JMS queue | Domain > Services > Messaging > JMS Modules > conversational-jms |
| | `wli.internal.b2b.rosettanetencoder.queue` | RosettaNet outbound JMS queue | Domain > Services > Messaging > JMS Modules > conversational-jms |
| | `wli.internal.sync2Async.soapResponse` | Asynchronous business process to synchronous client inbound queue | Domain > Services > Messaging > JMS Modules > conversational-jms |
| | `wli.internal.tracking.buffer` | Queue for process tracking | Domain > Services > Messaging > JMS Modules > conversational-jms |
| | `wli.internal.tracking.buffer_error` | Errors associated with queue for worklist timers | Domain > Services > Messaging > JMS Modules > conversational-jms |
| | `wli.internal.worklist.timer.queue` | Queue for worklist timers | Domain > Services > Messaging > JMS Modules > conversational-jms |

**Table D-1 WebLogic Integration Deployment Resources (Continued)**

| Resource Target | Resource | Description | Administration Console Navigation |
|---|---|---|---|
| | `wli.process.event.stream` | Queue for process events | Domain > Services > Messaging > JMS Modules > conversational-jms |
| | `wli.process.event.stream_error` | Errors associated with queue for process events | Domain > Services > Messaging > JMS Modules > conversational-jms |
| | `wlwJWSBuffer` | Buffering queue used by WLI Web Services Stack | Domain > Services > Messaging > JMS Modules > conversational-jms |
| | `cgJWSQueue` | Queue used by the WLI Web Services Stack | Domain > Services > Messaging > JMS Modules > conversational-jms |
| | `wlwJWSErrors` | Error queue used by WLI Web Services Stack | Domain > Services > Messaging > JMS Modules > conversational-jms |
| | `wli.b2b.failedmessage.queue` | B2B Queue | Domain > Services > Messaging > JMS Modules > conversational-jms |
| | `wli.internal.egmq.queue` | MQ EG Queue | Domain > Services > Messaging > JMS Modules > conversational-jms |
| | `wli.internal.egrdbms.queue` | RDBMS EG Queue | Domain > Services > Messaging > JMSM |
| Pinned Destinations (One managed server) | `wli.internal.scheduling.queue` | Timer queue for process archiving | Domain > Services > Messaging > JMS Modules > conversational-jms |

**Table D-1 WebLogic Integration Deployment Resources (Continued)**

| Resource Target | Resource | Description | Administration Console Navigation |
|---|---|---|---|
| | `wli.internal.scheduling.queue_error` | Error queue for process archiving | Domain > Services > Messaging > JMS Modules > conversational-jms |
| | `wli.internal.SQLStore.cleanup.documents` | Timer queue for doc store cleanup | Domain > Services > Messaging > JMS Modules > conversational-jms |
| | `wli.internal.egfile.queue` | Timer queue for file event generator | Domain > Services > Messaging > JMS Modules > conversational-jms |
| | `wli.internal.egmail.queue` | Timer queue for email event generator | Domain > Services > Messaging > JMS Modules > conversational-jms |
| | `wli.internal.egtimer.queue` | Timer queue for timer event generator | Domain > Services > Messaging > JMS Modules > conversational-jms |

## E



## F



## G



## H

## T



## W

## X