BEA WebLogic Integration™

Deploying WebLogic Integration Solutions

Version 8.1 Service Pack 6
Revised: June 2006

Copyright

Restricted Rights Legend

Trademarks or Service Marks

Contents

1. Introduction

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 solutions in a production environment. The following sections introduce key concepts and tasks for deploying WebLogic Integration in your organization:

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

This document focuses on the deployment phase of the WebLogic Integration software lifecycle. For detailed information describing the deployment portion of the software lifecycle for WebLogic Platform applications generally, see *Deploying WebLogic Platform Applications*.

For information about developing WebLogic Integration applications, see the documentation available at the following URL:

`http://edocs.bea.com/wli/docs81/index.html`

For examples of source and utilities to build, configure, and deploy WebLogic Integration applications, see the WebLogic Integration Solution Samples and the PO Sample that are available in the BEA dev2dev Code Library at the following URL:

`http://dev2dev.bea.com/code/wli.jsp`

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

Deployment Goals

WebLogic Integration 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 WebLogic Integration solutions, consider the following goals:

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

You can achieve these goals and others with every WebLogic Integration deployment.

Key Deployment Tasks

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

1. Define the goals for your WebLogic Integration deployment, as described in "Deployment Goals" on page 1-2.
2. Deploy WebLogic Integration 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 WebLogic Integration deployment. Chapter 3, "Understanding WebLogic Integration Clusters," provides descriptions of these components that will help you design the best possible environment for your application.
3. Deploy WebLogic Integration 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 WebLogic Integration deployment as described in Chapter 6, "Using WebLogic Integration Security."

For a detailed list of deployment tasks associated with WebLogic Platform applications, see Deployment Checklist in *Deploying WebLogic Platform Applications*.

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 a successful deployment requires input by people in each role.

Deployment Specialists

Deployment specialists coordinate the deployment effort. They are knowledgeable about the features of the WebLogic Integration product. They provide expertise in designing the deployment topology for an integration solution, based on their knowledge of how to configure various WebLogic Integration 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
- Process Application Resources
- Process Control Resources
- Message Broker Resources
- Event Generator Resources
- Trading Partner Integration Resources
- Application Integration Capabilities and Clients
- Relational Database Management System Resources
- Hardware, Operating System, and Network Resources

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 WebLogic Integration 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 WebLogic Integration solutions in any supported environment. The following sections describe the configurable WebLogic Server features that are most relevant to WebLogic Integration 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 topics:

- Introduction to WebLogic JMS in *Programming WebLogic JMS*
- JMS: Configuring and JMS: Monitoring in the WebLogic Server Administration Console Online Help

EJB Pooling and Caching

In a WebLogic Integration 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 WebLogic Integration performance by configuring the size of JDBC connection pools. A setting that is too low results in delays while WebLogic Integration waits for connections to become available. A setting that is too high results in slower DBMS performance.

For more information about WebLogic JDBC connection pools, see:

- "Overview of Connection Pools" in Introduction to WebLogic JDBC in *Programming WebLogic JDBC*.
- Default JDBC Configurations for WebLogic Platform.
- JDBC Connection Pools in the WebLogic Server Administration Console Online Help.

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 execute threads also determines the number of threads that read incoming socket messages (socket-reader threads). This number is, by default, one-third of the number of execute 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 "Creating Execute Queues" in Tuning WebLogic Server Applications in *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.

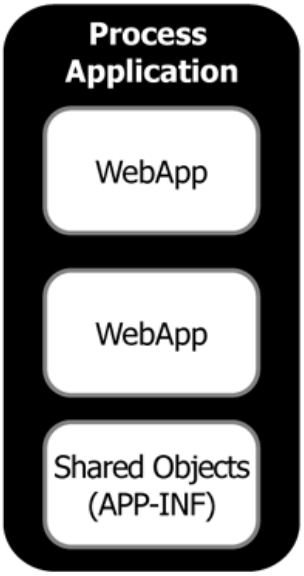
For information about the WebLogic J2EE-CA, see Overview of WebLogic J2EE Connectors in *Programming WebLogic Server J2EE Connectors*.

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 WebLogic Workshop application, as described in How Do I: Deploy a WebLogic Workshop Application to a Production Server? in WebLogic Workshop Help.

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, as shown in the following figure.

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 WebLogic Integration 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 WebLogic Workshop SOAP/JMS protocol or the WebLogic Integration JMS Event Generator.

  For more information about the SOAP/JMS protocol, see Building a JMS Client in WebLogic Workshop Help. For more information about the JMS Event Generator, see Event Generators in *Managing WebLogic Integration Solutions*.

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

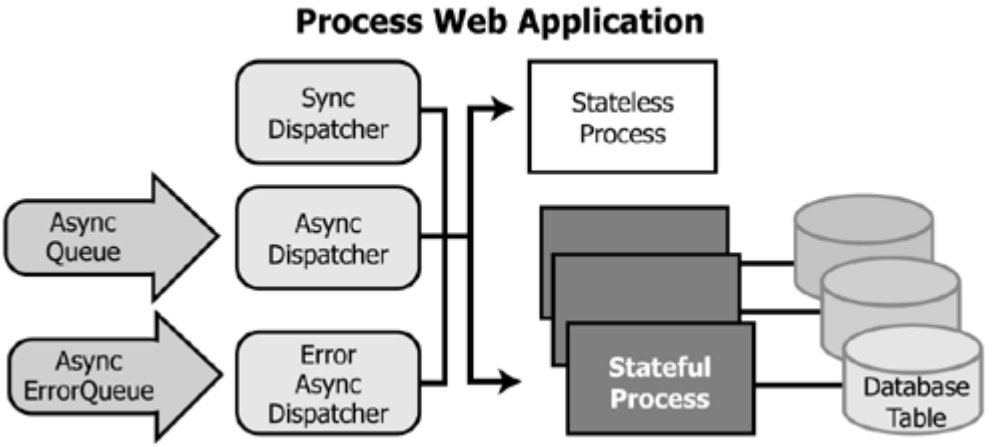
  This is used for dispatching exception elements in process definition.

- A pool of stateless session beans for each stateless process

- A pool of entity beans for each stateful process

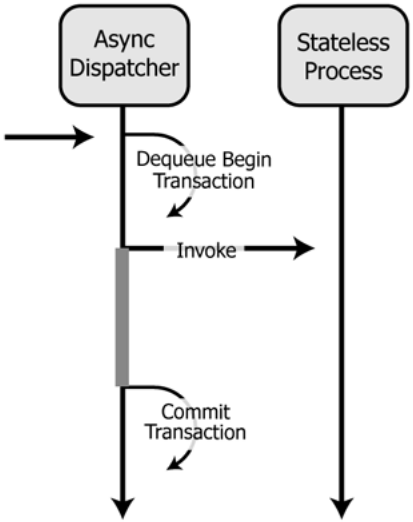
The following figure shows the components in a process web application.

Figure 1-2 Process Web Application



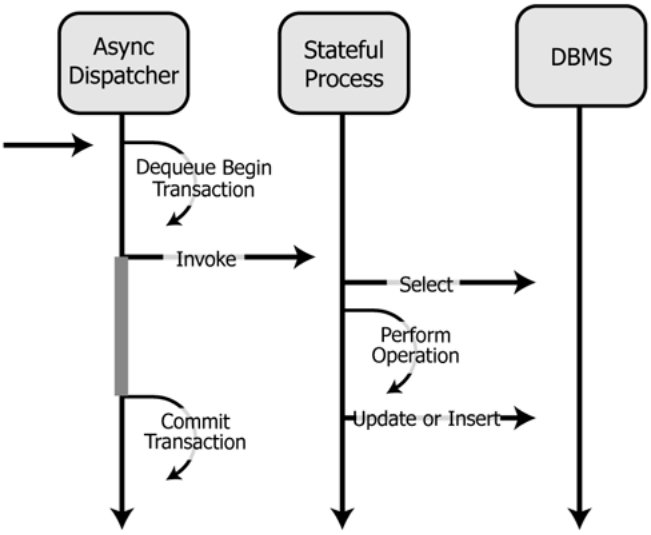
The asynchronous dispatcher has different interactions with stateless and stateful processes. For an illustration of the interaction between the asynchronous dispatcher and a stateless process, see the following figure.

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 the following figure.

Figure 1-4 Interaction Between Dispatchers and a Stateful Process



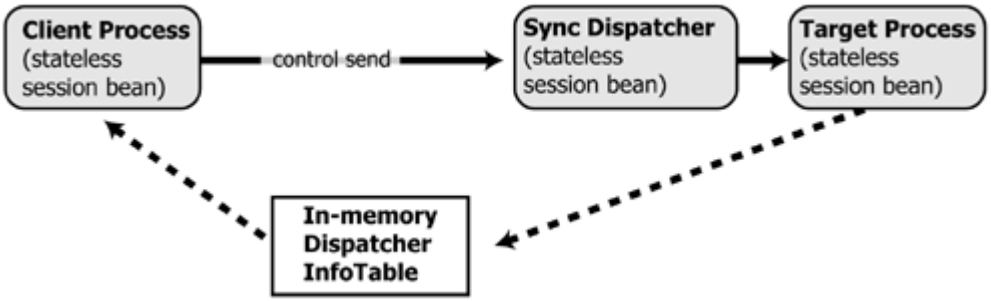
Process Control Resources

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 will stay 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.

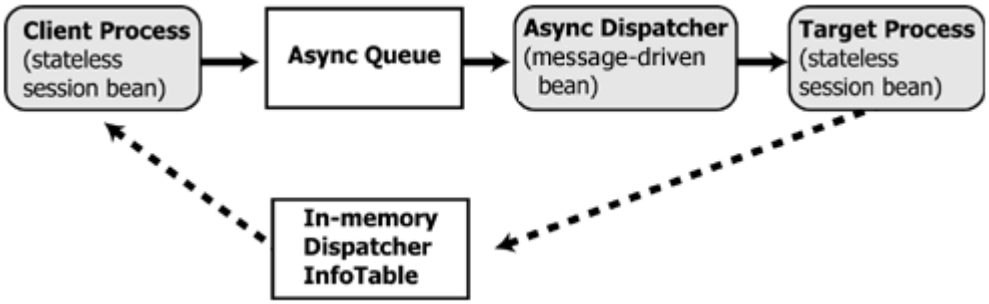
The behavior of a process call depends on whether it is being used for a synchronous or asynchronous dispatch. The following figure shows the behavior of a process control used for a synchronous dispatch.

Figure 1-5 Process Control Used for a Synchronous Dispatch



The following figure 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 will be consumed from the default thread pool. For more information about execution thread pools, see "Creating Execute Queues" in Tuning WebLogic Server Applications in *WebLogic Server Performance and Tuning*.

For more information about this configuration, see Synchronous Clients for Asynchronous Business Processes in "Building Synchronous and Asynchronous Business Processes" in the

"Guide to Building Business Processes" in "Building Integration Applications" in the WebLogic Workshop Help. For implementation examples, see the WebLogic Integration Solution Samples.

Message Broker Resources

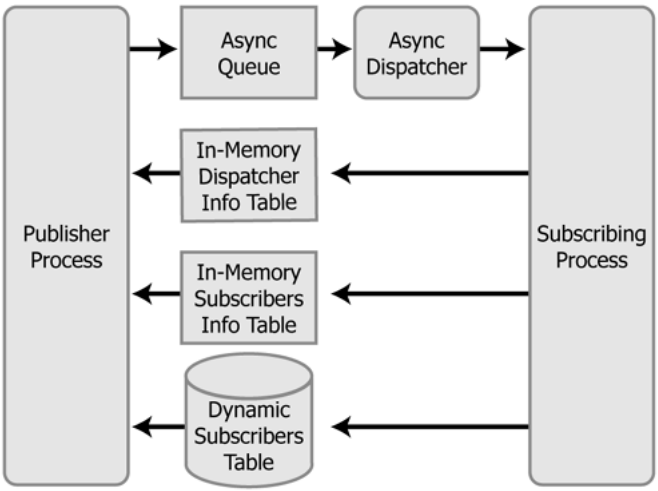
Any time the Message Broker publishes a message through a Message Broker publish control or event generator, the following actions occur:

- 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 will be 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 the following figure.

Figure 1-7 Publishing Using the Message Broker Publish Control



Event Generator Resources

WebLogic Integration has a number of native event generators:

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

In addition, event generators are used for application integration EIS events. (For additional information regarding event generators for EIS adapters, see "Application Integration Capabilities and Clients" on page 1-19.)

The 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 to either a single

managed server when using a physical JMS destination, or to the cluster when using distributed destinations.

The File, Email, and Timer Event Generators

These event generators are "polling" event generators, in that they 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 the following table.

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 to the cluster.

Note: WebLogic Integration supports polling event generators targeted to a single server. However, this configuration is not appropriate for applications that require high availability.

The MQSeries and HTTP Event Generators

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. 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 the HTTP and MQSeries event generators can be targeted to a single managed server or cluster, as described in the Event Generators section of *Managing WebLogic Integration Solutions*.

The 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, as described in the Event Generators section of *Managing WebLogic Integration Solutions*.

Suspending Event Generators

It is important to note that 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 the Event Generators section of *Managing WebLogic Integration Solutions*.

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 WebLogic Integration releases. The Trading Partner Integration resources currently retain B2B as part of their names

When you deploy WebLogic Integration 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 *Managing WebLogic Integration Solutions*.

Trading Partner Management Repository

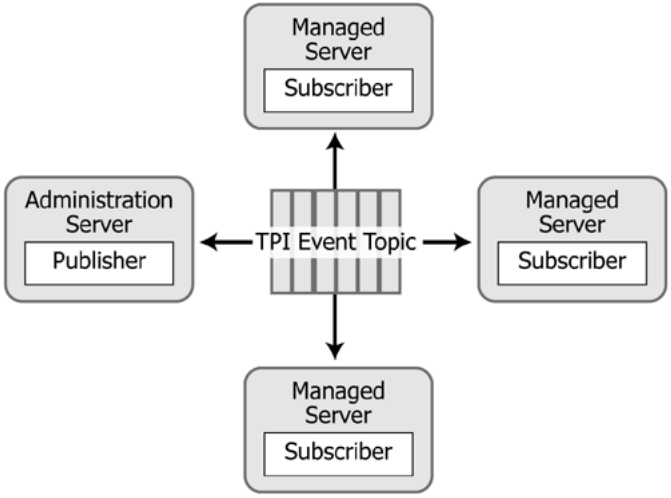
The Trading Partner Management Repository is an important part of Trading Partner Integration. Database operations on this repository and for all of Trading Partner Integration are performed through the `JDBCTxDataSource` named `cgDataSource` using the `JDBCPool` named `cgPool`.

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 8.1 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 the following figure.

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 *Managing WebLogic Integration Solutions*.

Trading Partner Integration Initialization and Run-Time Operations

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

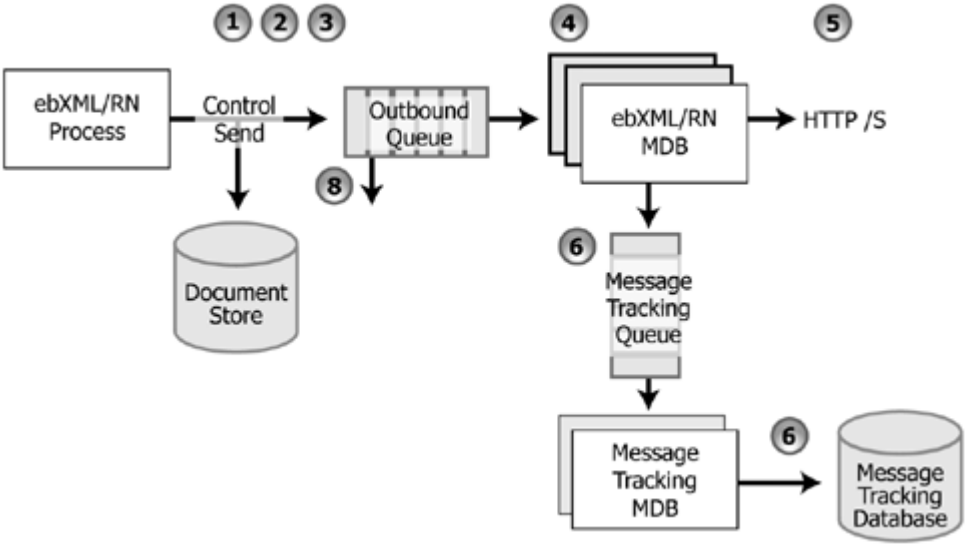
Warning: The WLI-B2B Startup EJB has an `initial-beans-in-pool` setting of 1. Changing this value will cause Trading Partner Integration startup to fail.

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

The following figure shows the path of an outgoing business message.

Figure 1-9 Outgoing Message Path



The preceding figure illustrates the following process flow:

1. A message is sent from a WebLogic Integration 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 WebLogic Integration 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.

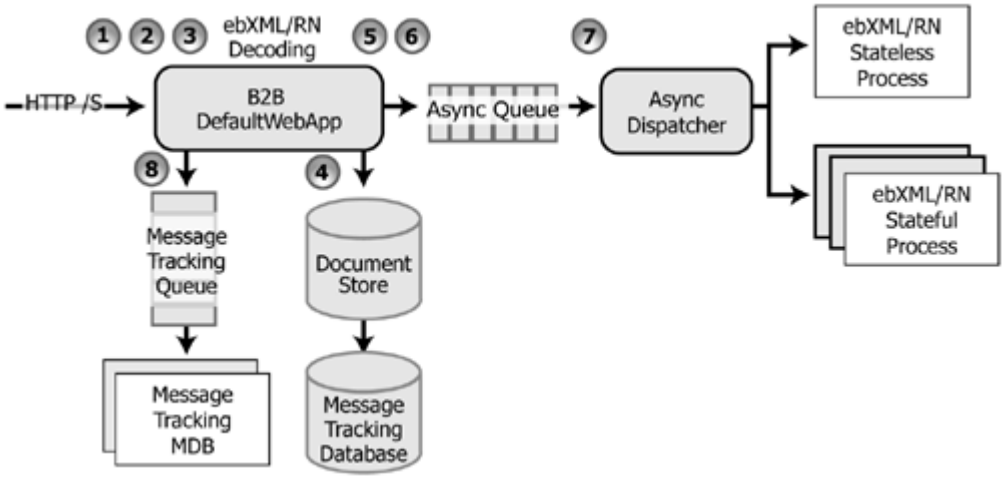
4. Each of these queues has its own message-driven bean(s) listening to the queue. The message-driven beans are WLI-B2B RosettaNet for RosettaNet and WLI-B2B ebXML for ebXML. The pool size of these message-driven beans can be increased as needed to support customer environments that experience high message 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 *Managing WebLogic Integration Solutions*.

Incoming Messages

The following figure 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 *Managing WebLogic Integration Solutions*.

Application Integration Capabilities and Clients

The following sections describe the major capabilities of WebLogic Application Integration and how clients make use of those capabilities:

- Synchronous Service Invocations
- Asynchronous Service Invocations
- Events

For information about clustering and application integration, see Chapter 3, "Understanding WebLogic Integration Clusters."

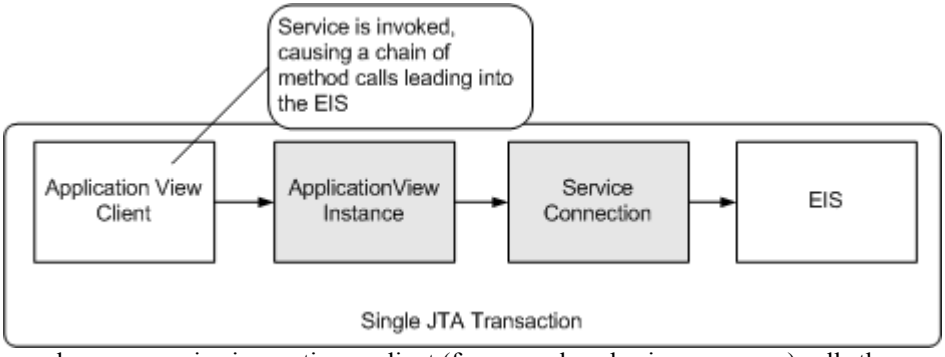
Application integration functionality is integrated in the WebLogic Integration product. In order to use application integration with other components of the WebLogic Platform (such as WebLogic Workshop or WebLogic Portal) you must configure a domain that includes each of those components. For information about creating domains, see Creating a New WebLogic Domain in *Using the Configuration Wizard*. For more information about deploying WebLogic Platform applications, see *Deploying WebLogic Platform Applications*.

Synchronous Service Invocations

Use synchronous invocations when the underlying EIS can respond quickly to requests, or when the client application can afford to wait.

The following figure illustrates the flow of a synchronous service invocation.

Figure 1-11 Synchronous Service Invocations



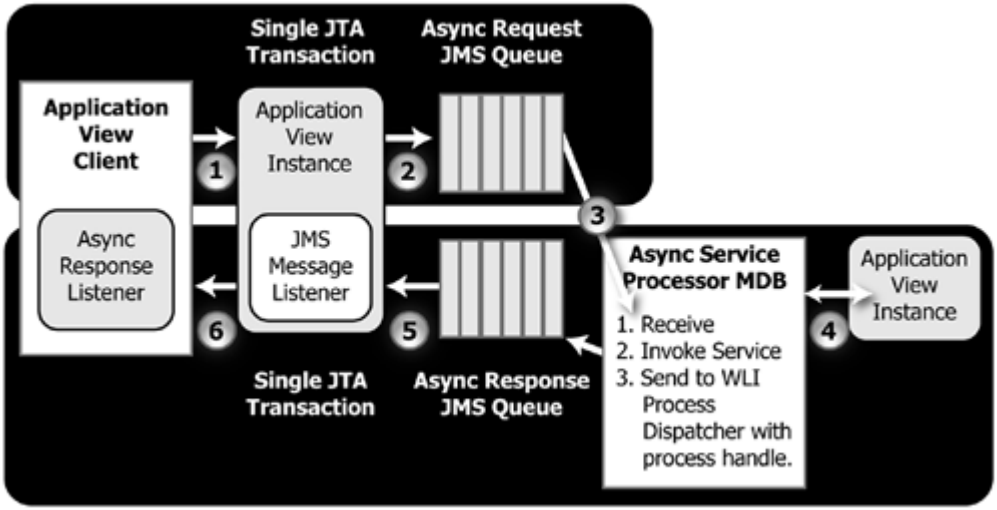
In a synchronous service invocation, a client (for example, a business process) calls the application view (for example, as a control in the business process). The application view calls the adapter using a synchronous Common Client Interface (CCI) request. The service adapter is a J2EE-CA service adapter that actually processes the request.

Note: When a process acts as a client to an EIS, the process is stalled while it waits for the request to complete, tying up a WebLogic execute thread, business process, EJB instances, and other resources. To optimize throughput, consider using asynchronous invocations instead unless the underlying EIS system can respond quickly to the request.

Asynchronous Service Invocations

The following figure illustrates asynchronous service processing in WebLogic Integration.

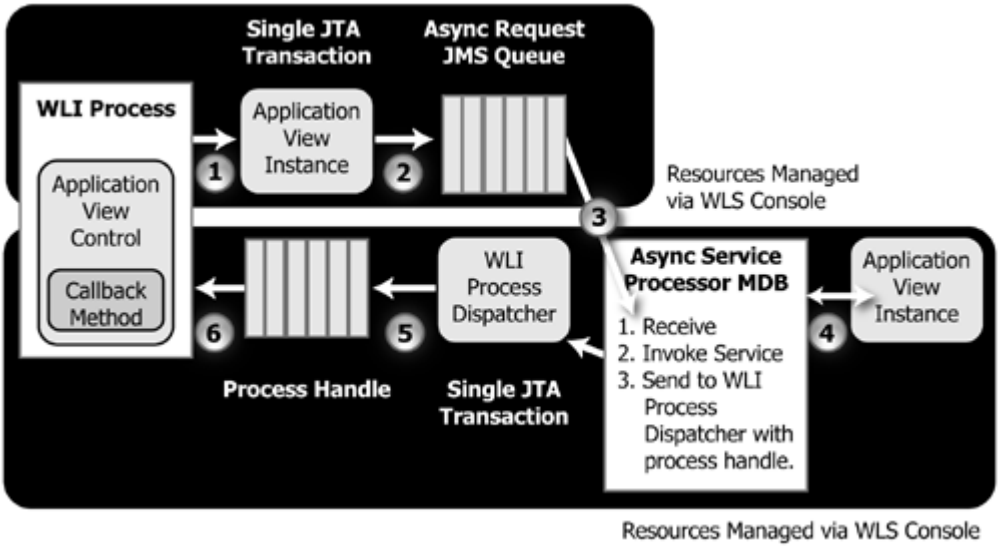
Figure 1-12 Asynchronous Service Invocations—Programmatic/Custom Client



Note: When using the Application View control from a WebLogic Integration process or WebLogic Workshop Web service, the asynchronous response is delivered directly to the EJB for the process or Web service without being posted to the asynchronous response JMS queue.

The following figure shows the processing that takes place during asynchronous service invocations in a WebLogic Integration process client.

Figure 1-13 Asynchronous Service Invocations—WebLogic Integration Process Client



The preceding diagram illustrates the following process flow:

1. An application view client instantiates an application view instance:
   - In the case of a WebLogic Integration process (described in the preceding graphic), the ApplicationView Instance is encapsulated in an ApplicationView Control instance. Asynchronous requests are tagged with a process handle that allows the AsyncServiceProcessor to route the response back to this process instance.
   - In the case of a programmatic/custom client (described in the figure titled "Asynchronous Service Invocations—Programmatic/Custom Client" earlier in this section), the *ApplicationView Instance* is used directly.
2. The application view instance creates an `AsyncServiceRequest` object and sends it to the `wli.internal.ai.async.request` queue.
3. The `AsyncServiceProcessor` message-driven bean receives the message from the queue in a first in, first out (FIFO) manner. The `AsyncServiceProcessor` uses the `AsyncServiceRequest` object in the JMS ObjectMessage to determine the qualified name, service name, request document, and response destination for the application view.
4. The `AsyncServiceProcessor` uses an application view EJB to invoke the service synchronously. The service is translated into a synchronous CCI-based request/response message for the resource adapter.

5. The `AsyncServiceProcessor` receives the response. The response is subsequently encapsulated into an `AsyncServiceResponse` object and sent to the response destination provided in the `AsyncServiceRequest` object:
   - In the case of a WebLogic Integration process client, the response is dispatched back to the process handle tagged to the request.
   - In the case of a programmatic/custom client, the response is placed onto the `wli.internal.ai.async.request` queue.
6. The client receives the asynchronous response:
   - In the case of a WebLogic Integration process client, the response is delivered via a callback (`on`*`ServiceName`*`Response`) on the `ApplicationView` control instance.
   - In the case of a programmatic/custom client, the client receives the `AsyncServiceResponse` message as a JMS ObjectMessage and passes it to the `AsyncServiceResponseListener` supplied in the `invokeServiceAsync()` call shown in step 2.

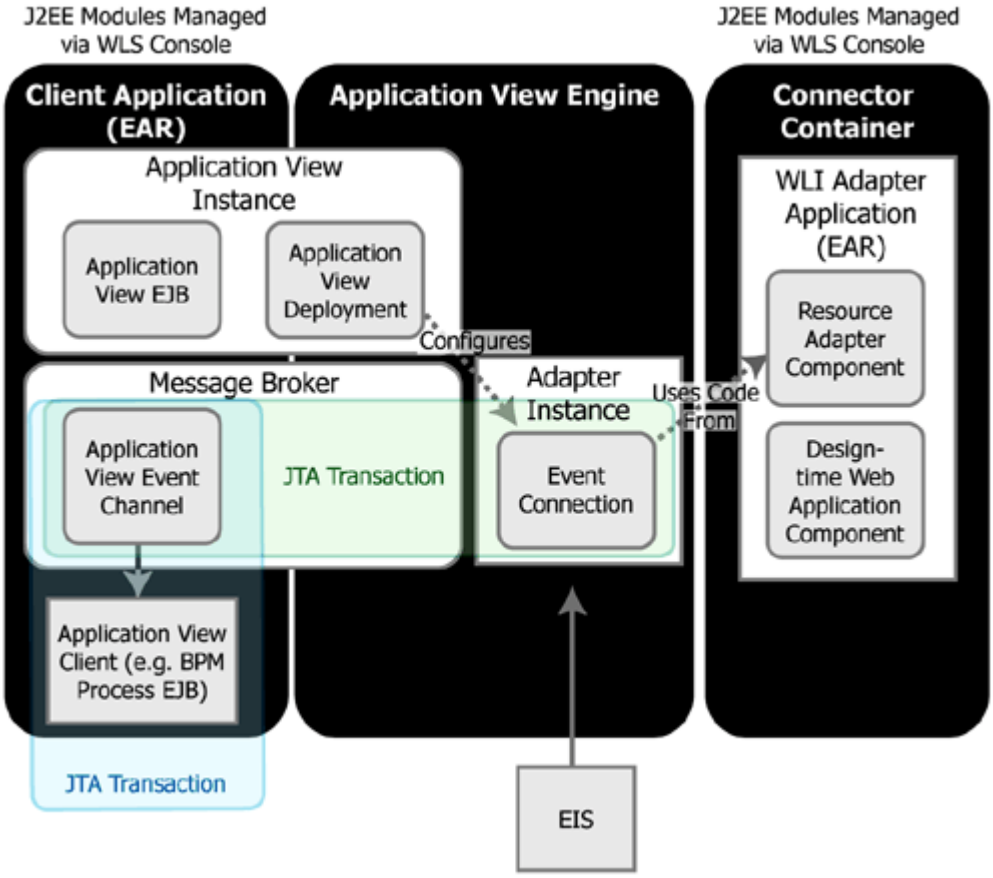
Events

Application integration adapters generate events that are consumed by business processes or Web services. Events are then delivered to application view clients through one of two methods:

- Events are delivered to WebLogic Integration process clients through a `MessageBroker` event channel (named for the `ApplicationView` and event type for the event).
- In the case of a programmatic/custom client, events are delivered through an `EventListener` object provided by the client. The event is actually delivered to the `EventListener` through a JMS topic and JMS MessageListener instance (provided by the `ApplicationView` instance) on that topic.

The following figure illustrates event processing in WebLogic Integration.

Figure 1-14 Events via Message Broker



The preceding figure illustrates the following sequence of steps for event processing:

1. An event occurs in an enterprise information system (EIS).
2. The event data is transferred to the event generator in the adapter instance. The details of this transfer and the data transferred are adapter-specific. The event generator transforms the EIS-specific event data into an XML document and posts an IEvent object to the embedded event router instance in the adapter instance. The event generator and embedded event router instance together constitute the event connection for an adapter instance.

3. The event connection passes the `IEvent` object to all registered `ApplicationView` event subscription objects that have indicated interest in this event type. Subscription objects are registered at application view deployment time.
4. The event subscription object delivers the `IEvent` object to both the `MessageBroker` event channel for the event type (named for `ApplicationView`/event) and the `wli.internal.ai.event` JMS topic.
5. Clients receive the event by one of two methods:
   - WebLogic Integration process clients receive the event through the event channel.
   - Programmatic/custom clients receive the event through an `EventListener` instance provided by the client. The `EventListener` instance is called from a JMS message listener registered on the event topic.

Relational Database Management System Resources

WebLogic Integration 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 WebLogic Integration performance. For information about database tuning requirements associated with WebLogic Integration 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 WebLogic Integration performance. Deployments must comply with the hardware and software requirements described in the *WebLogic Integration 8.1 Release Notes*.

CHAPTER 2

Configuring a Single-Server Deployment

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

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

Step 1. Configure a Database

Configure one of the following databases for your domain:

- IBM DB2
- Microsoft SQL Server
- Oracle
- Sybase

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

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 WebLogic Integration applications, see Preparing Your Database and the database-specific notes in Maintaining Availability.

For the latest information about specific databases, see the *WebLogic Integration 8.1 Release Notes*.

Step 2. Prepare a WebLogic Integration Domain

To prepare a WebLogic Integration environment, complete the tasks described in the following sections:

- Creating a WebLogic Integration Domain Using the Configuration Wizard
- Creating the Database Tables

Creating a WebLogic Integration Domain Using the Configuration Wizard

You begin the definition of a WebLogic Integration 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 *Using the Configuration Wizard*.

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

1. From the Start Menu, choose **Programs—BEA WebLogic Platform 8.1—Configuration Wizard**.

   The Domain Configuration Wizard is launched. It prompts you for data with which to configure your domain.

2. Respond to the Domain Configuration Wizard prompts by providing the information described in the following table.

In this window . . .	Perform the following action . . .
Create or Extend a Configuration	Select Create a new WebLogic configuration.
Select a Configuration Template	Select the Basic WebLogic Integration Domain template
Choose Express or Custom Configuration	Select Custom.
Configure the Administration Server	Accept all the default values. **Note:** When you configure the administration server, we recommend that you accept the default Server Name (`cgServer`), as prompted by the Domain Configuration Wizard.
Managed Servers, Clusters, and Machine Options	Do you want to distribute your WebLogic configuration across managed servers, clusters and physical machines? Accept the default value (No). **Note:** A WebLogic Integration domain that includes an administrative server and one or more managed servers must include a cluster. A WebLogic Integration domain that includes an administrative server and one or more managed servers without a cluster is an unsupported configuration.
Database (JDBC) Options	Do you want to define JDBC components, such as Connection Pools, Data Sources, and MultiPools? Select Yes.

Configure JDBC Connection Pools	Configure the JDBC Connection Pools for WebLogic Integration. For a description of the correct JDBC settings for an XA domain, see "How Do I: Create a WebLogic Integration Domain Using Oracle with an XA Driver?" in How Do I ...? in *Using the Configuration Wizard*. You must configure `cgPool` for WebLogic Integration to function. If the reporting data tables are referenced through a different data store than `cgPool`, then you must define `bpmArchPool` as a valid pool and configure it as the Reporting Data DataStore in the WebLogic Integration Administration Console. These configuration changes take effect on the next start of WebLogic Server. (For information on configuring the Reporting Data DataStore in the WebLogic Integration Administration Console, see "Configuring the Archive Data Store" in System Configuration in *Managing WebLogic Integration Solutions*.) You can choose additional Connection Pools for application use. If you are going to define a JDBC datastore, you must define at least one non-XA JDBC connection pool. JDBC datastores work only with non-XA JDBC connection pools. **Note:** WebLogic Integration may not be certified with all drivers. For a list of certified drivers, see "Supported Databases and Drivers" in Supported Configurations for WebLogic Platform in *Supported Configurations*.
Assign JDBC Connection Pools to MultiPools	Configure any Multipools needed by the application. Multipools are not required for the operation of WebLogic Integration itself.
Configure JDBC Data Sources	Accept the defaults for `cgDataSource` and `bpmArchDataSource`. Add any application-specific data sources. **Note:** No more than one non-XA datasource can be used in the same transaction. Data sources cannot share an XA JDBC connection pool.
Test JDBC Connection Pools and Setup JDBC Database	Optional test. For information about loading WebLogic Integration tables in the JDBC database, see "Creating the WebLogic Integration Tables" in Configuring a Production Database in *Managing WebLogic Integration Solutions*.
Messaging (JMS) Options	Do you want to define JMS components, such as Stores, Topics, and Queues? Select Yes. **Note:** Exercise caution in changing JMS option settings. Inappropriate settings may cause WebLogic Integration to function unpredictably.
Configure JMS Connection Factories	Accept the defaults for `cgQueue` and WLI-B2B System Topic Factory.

Configure JMS Destination Key(s)	Configure any keys needed for the application. None are needed for the operation of WebLogic Integration.
Configure JMS Template(s)	Configure any templates needed for the application. None are needed for the operation of WebLogic Integration.
Configure JMS File Stores	Accept the defaults for `rmfilestore` and add any file stores needed by the application.
Configure JMS JDBC Store	Select the appropriate connection pool from the list, then add any JDBC stores needed by the application.
Configure JMS Servers	WebLogic Integration System Queues can be mixed with Application Queues within the same JMS Server.
Assign JMS Servers to WebLogic Servers	Accept the defaults.
Configure JMS Topics	Accept the defaults, and add any user-defined topics.
Configure JMS Queues	Accept the defaults, and add any user-defined queues.
Configure JMS Distributed Topics	Accept the defaults, and add any user-defined distributed topics.
Configure JMS Distributed Queues	Accept the defaults, and add any user-defined distributed queues.
Assign JMS Distributed Destinations to Servers or Clusters	Accept the defaults.
Configure JMS Distributed Topic Members	Accept the defaults. There should be members for each configured managed server.
Configure JMS Distributed Queue Members	Accept the defaults.
Applications and Services Targeting Options	Do you want to target servers and clusters onto which Applications, JMS component services, JDBC component services, and other services are deployed? Select Yes.
Target Applications to Servers or Clusters	Accept the defaults.

Target Services to Servers or Clusters	Accept the defaults for non-XA data sources. Configure the appropriate target for XA data sources. For more information, see "Creating XA Domains Using Configuration Templates" in How Do I... in *Using the Configuration Wizard*.
Configure Administrative Username and Password	Select user names and passwords.
Configure Windows Options	Define operating system configuration information for Windows platforms. **Note:** Windows platform users only will encounter this page while using the Configuration Wizard. Configuring Windows options is not a requirement of a cluster domain configuration.
Configure Server Start Mode and Java SDK	Select Production Mode, and then select either the Sun SDK or JRockit SDK.
Create WebLogic Configuration	Select the name of your custom domain.

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

Your WebLogic Integration domain includes two configuration files:

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

For more information about `config.xml`, see *WebLogic Server Configuration Reference*.

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

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*.

Creating the Database Tables

When preparing a production environment (working in *noniterativedev* mode), you must create the WebLogic Integration database tables. For the procedure to create these tables, see "Creating

the WebLogic Integration Tables" in Production Database in *Managing WebLogic Integration Solutions*.

For more information about conversational state database tables, 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*.

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 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 Server

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

- Starting the Server
- Monitoring and Shutting Down Your Server

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 Server

To start the server, 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 "Starting the Administration Console" in Overview of WebLogic Server System Administration in Configuring and Managing WebLogic Server.

2. In the WebLogic Server Administration Console navigation tree, select the name the server.

3. Select the **Control** tab.
4. Click **Start this Server**.

   For more information about starting the server, see Starting and Stopping Servers in *Adminstration Console Online Help*.

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 Monitoring a WebLogic Server Domain in Configuring and Managing WebLogic Server. For information about monitoring your WebLogic Integration domain, see "Run-Time Tuning Issues" in Performance Tips in the WebLogic Integration *Solutions Best Practices FAQ*.

If you need to shut down your WebLogic Integration 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 WebLogic Integration.

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

Step 5. Deploy a WebLogic Integration Application

Once you have configured and secured your WebLogic Integration 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 WebLogic Integration application.

If you did not configure all the queues necessary for your application while configuring your WebLogic Integration domain as described in "Creating a WebLogic Integration Domain Using the Configuration Wizard" on page 2-2, you can configure them now using the WebLogic Server Administration Console.

Note: Async request and async request error queues, as well as database conversational state tables, are created automatically for applications in the WebLogic Workshop development environment. You must create these queues and tables manually for production environments.

For information on how to configure 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 on how to configure JMS resources using the WebLogic Server Administration Console, see "How Do I: Deploy WebLogic Workshop Web Services to a Production Server?" in How Do I... in WebLogic Workshop Help.

Note: If your WebLogic Integration 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 3-16.

You can update environment-specific information in your Application Views and adapter instances either before or after deploying your WebLogic Integration application:

- To update an unbooted domain before deployment, use the aiConfigurator as described in Appendix B, "Administering Environment-Specific Application Integration Information."
- To update after deployment (and redeploy, as necessary), use the WebLogic Integration Administration Console as described in Application Integration in *Managing WebLogic Integration Solutions.*

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

For examples of automation scripts that build, configure, and deploy WebLogic Integration applications outside of an interactive console environment, see the WebLogic Integration Solution Samples and the PO Sample that are available in the BEA dev2dev Code Library at the following URL:

`http://dev2dev.bea.com/code/wli.jsp`

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*.

Step 6. Update Your Domain as Your Production Environment Changes

Production environments change over time and as application use increases. This section describes how to update your domain in response to common production environment change scenarios:

- Changing an EIS Instance
- Installing a New Version of Your Application

For information about promoting WebLogic Platform applications from development environments to production environments, see "Steps to Promote WebLogic Platform Applications" in Overview of WebLogic Platform Development in *Deploying WebLogic Platform Applications*.

Changing an EIS Instance

EIS instances are sometimes phased out, and new instances (possibly with new versions of EIS software, new hardware, and so on) are brought online. When this happens, WebLogic Integration administrators need to gracefully transition application views and adapter instances to the new EIS instance, and away from the old.

This situation is similar to an EIS instance failure, but not as urgent. In this case, you suspend the application views and adapter instances pointing at the old EIS instance, and then follow the instructions in "EIS Instance Failover" on page 5-7 to point the application views and adapter instances to the new EIS instance.

Installing a New Version of Your Application

As your business requirements change, you may need to develop and deploy new versions of your WebLogic Integration application. To install a new version of your WebLogic Integration application, perform the following steps:

1. Delete the application from WebLogic Server.
2. Deploy the new version of the application.

You perform the preceding steps using the WebLogic Server Administration Console. For more information, see "Configuring and Deploying a New Enterprise Application or Web Service" in Enterprise Applications in the *Adminstration Console Online Help*.

CHAPTER 3

Understanding WebLogic Integration Clusters

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

- Understanding WebLogic Integration Clusters
- Designing a Clustered Deployment
- Load Balancing in a WebLogic Integration Cluster
- High Availability in a WebLogic Integration Cluster
- Deploying Adapters
- Deploying Event Generators

For general information on WebLogic Platform applications in cluster environments, see Understanding the Target Environment in *Deploying WebLogic Platform Applications*.

Understanding WebLogic Integration Clusters

Clustering allows WebLogic Integration 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. WebLogic Integration provides load balancing so that resource requests are distributed proportionately across all machines. A WebLogic Integration 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 WebLogic Server cluster domain consists of only one administration server, and one or more managed servers. The managed servers in a WebLogic Integration domain can be grouped in a cluster. When you configure WebLogic Integration 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: WebLogic Integration domains can support multiple clusters. However, you must still target WebLogic Integration system resources and applications on a single WebLogic Integration cluster in a multiple cluster domain.

The topics in this section provide the information you need to configure WebLogic Integration in a clustered environment. Although some background information about how WebLogic Server supports clustering is provided, the focus is on procedures that are specific to configuring WebLogic Integration 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 Tuning WebLogic Server in *WebLogic Server Performance and Tuning*

Designing a Clustered Deployment

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

Introducing WebLogic Integration 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 Reference in *Using the Configuration Wizard*. For information about creating WebLogic Integration domains using the Domain Configuration Wizard, see Creating a New WebLogic Domain in *Using the Configuration Wizard*.

The domain you create with the Configuration Wizard must have one and only one target which will contain WebLogic Integration 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 WebLogic Integration target defaults to the first WebLogic Integration cluster or, in the absence of a cluster, the first managed server. For more information about the `wli-config.properties` file, see Appendix C, "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 general 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*.

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 WebLogic Integration deployment such that the cluster domain equals the management and security domain.

Deploying WebLogic Integration 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 WebLogic Integration domain using the Configuration Wizard. You can also configure these attributes manually using the Servers node in the WebLogic Server Administration Console.

For a list of configurable WebLogic Integration deployment resources, see Appendix D, "WebLogic Integration Deployment Resources." It describes the default targeting of each resource in a clustered WebLogic Integration 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 WebLogic Integration deployment configuration requirements:

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

Two-Phase Deployment of WebLogic Integration

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

Listing 3-1 Deploying the WebLogic Integration 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 Recovering Failed Servers in Configuring and Managing WebLogic Server.

Load Balancing in a WebLogic Integration Cluster

One of the goals of clustering your WebLogic Integration 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 WebLogic Integration cluster:

- Load Balancing HTTP Functions in a Cluster
- Load Balancing JMS Functions in a Cluster
- Load Balancing Application Integration 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 WebLogic Integration or WebLogic Integration 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 JMS: Tuning in the WebLogic Server Administration Console Online Help.

Load Balancing for Synchronous Clients and Asynchronous Business Processes

If your WebLogic Integration 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`.

Load Balancing Application Integration Functions in a Cluster

Application integration allows for load balancing of both services and events within a cluster. Each type is discussed in a following section.

Synchronous Services

Synchronous services are implemented as method calls on a session EJB. As such, they will be load balanced within the cluster according to EJB load balancing rules. After being published at design-time, each application view is represented as two session EJBs: one stateless and one stateful.

In normal operation, the services are invoked using the stateless session EJB, and thus load balancing will occur on a per-service basis. This means that each time you invoke a service on an application view, you may actually be routed to a different EJB on a different WebLogic managed server instance.

When using the local transaction facilities of the application view and during a local transaction, services are invoked using the stateful session EJB. The stateful session EJB is used to hold open the connection to the EIS, so that the local transaction state can persist between service invocations. In this mode, service invocations become pinned to a single EJB instance on a single managed server within the cluster. Once the local transaction completes (either through a commit or rollback), normal per-service load balancing applies.

Asynchronous Services

Asynchronous services are always invoked as method calls on a stateless session EJB. You cannot use the local transaction facility of the application view for asynchronous service invocations.

A single asynchronous service invocation translates to two method invocations on up to two different stateless session EJB instances. Thus, load balancing for asynchronous service occurs on two occasions: upon receipt of the request, and in the execution of the request and delivery of the response.

In addition, both the asynchronous service request and response are posted to a distributed JMS queue. As a result of this, JMS load balancing applies to both the request and response. This means that the `invokeServiceAsync` method of the application view may be serviced on one

managed server, the request delivered to a second managed server where the request is processed and the response generated, and the response delivered to a third server for retrieval by the client.

Note: When using the application view control from a WebLogic Integration process or WebLogic Workshop Web service, the asynchronous response is delivered directly to the EJB for the process or Web service without being posted to the asynchronous response JMS queue.

Application Integration Events

Application integration adapters generate events that are consumed by WebLogic Integration processes or by WebLogic Workshop Web services. Events are generated inside an EIS instance outside the WebLogic Integration cluster. Application integration is made aware of these events by event objects created by event connections within adapter instances. Those adapter instances reside on individual managed servers within the cluster.

The behavior of event delivery from the point of origin in the EIS to the point where they are handed off to application integration is adapter-specific and not defined by application integration. However, once an event generator has delivered the event into application integration, the event is load balanced for delivery to clients on any managed server in the cluster.

The behavior of event delivery within a cluster depends on whether or not the following conditions exist:

- The adapter for the EIS supports multiple event connections within a cluster for a single application view or event type
- The event connection delivering the events has been deployed to more than one managed server in the cluster

If both of these conditions exist, event delivery and the subsequent processing of those events by processes and Web services will scale as the number of managed servers in the cluster scales.

Support for multiple event connections within a cluster for a single application view or event type depends on the design of the adapter. For example, the DBMS sample adapter included in the Adapter Development Kit does provide this support. Consult your adapter vendor or adapter documentation to determine if the adapter you are using provides this support.

Deployment of the event connection to more than one managed server in the cluster depends on how you have configured the event connection and adapter instance. Using the WebLogic Integration console, WebLogic Integration administrators can target an event connection at zero or more managed servers in the cluster. If the adapter supports multiple event generators in a cluster, it is best practice to deploy the event connection to all managed servers in the cluster.

For information about how WebLogic Integration processes adapter events, see "Events" on page 1-23.

High Availability in a WebLogic Integration Cluster

Message-driven beans consume messages from JMS destinations. A number of message-driven beans are deployed on each WebLogic Integration destination. For a complete list of WebLogic Integration destinations (JMS queues and topics), see the resource type of Services—JMS in Table D-1, "WebLogic Integration Deployment Resources," on page D-2.

Highly Available JMS for Application Views

The ability to configure multiple physical destinations as members of a single distributed destination set provides a highly available implementation of WebLogic JMS. Specifically, for each node in a cluster, an administrator should configure one physical destination for a distributed destination. If one node in the cluster fails, making the physical destination for that node unavailable, then other physical destinations configured as members of the distributed destination can provide service to JMS producers and consumers. (This is the way the Configuration Wizard generates domains for a cluster.)

Message-driven beans consume messages from distributed destinations. Distributed destinations contain one physical destination for each instance of WebLogic Server. A single message producer on a distributed queue is bound to a single physical destination. Message-driven beans are bound to the physical destination in the server on which they are deployed (server affinity).

When a managed server fails in a cluster, the message-driven beans from the failed server are migrated atomically, but not automatically, to prevent multiple message processing. In the case of those destinations that must be deployed as singletons in a clustered environment, high availability is still achieved because a JMS server and all of its destinations can be migrated to another WebLogic Server within a cluster.

The following sections describe examples of how WebLogic Integration uses distributed destinations and server affinity to achieve high availability in a clustered deployment:

- High Availability for Asynchronous Service Requests to Application Views
- High Availability for Event Delivery from Application Views

High Availability for Asynchronous Service Requests to Application Views

Application integration uses a distributed JMS queue (`wli.internal.ai.async.request`) to manage asynchronous requests entered by application view clients. The Asynchronous Service

Request Processor is the message-driven EJB that processes these requests, and this bean is deployed to all servers in a cluster. The method that application integration uses to return the asynchronous response to the application view client depends on the type of the client that submitted the request:

- For clients submitting a request directly from the application view client, application integration posts responses to a distributed JMS queue (`wli.internal.ai.async.response`).
- For clients submitting a request using the application view control from a WebLogic Integration process or WebLogic Workshop Web service, application integration delivers responses using the WebLogic Workshop service callback dispatching mechanism. This dispatch is a sequence of synchronous method calls encapsulated in a single JTA transaction.

If a physical queue fails before an asynchronous service request is received by a message-driven bean, the request is unavailable until the physical queue returns to normal operation. The same scenario is true for asynchronous service responses. In the event of a managed server failure, the messages being managed by the JMS server on that managed server become unavailable. Messages being managed by JMS servers on other managed servers remain available and will be processed normally.

Note: Asynchronous service requests that have not yet been enqueued to the asynchronous request queue will be lost if the managed server on which the application view instance resides fails.

All asynchronous requests registered on the asynchronous request queue are dequeued and processed in the scope of a JTA transaction. Applications requiring highly available processing of requests should use the ApplicationView control to invoke the asynchronous service. This will cause the asynchronous response to be delivered to the containing process and web service instance in the same transaction that dequeued the asynchronous request and processed that request against the EIS.

When a server fails, asynchronous requests being processed at that time will be rolled back onto the request queue. When WebLogic transaction manager performs transaction recovery for the failed server, any XA-capable resource managers that have been used by the application view will be asked to rollback their work.

WebLogic Integration process client service invocations made through the application view control occur in a JTA transaction. When a managed server fails, the process will be rolled back to the last commit point. If the process was started using the MessageBroker or other persistent message system, then the entire process can be retried after the start message is delivered.

Application view clients using an `AsyncServiceResponseListener` instance are actually receiving those messages using the JMS client acknowledge mode. If the server hosting the asynchronous response fails, the message is not acknowledged and will remain on the server.

For information about processing of synchronous and asynchronous invocations for application integration functions, see "Application Integration Capabilities and Clients" on page 1-19.

High Availability for Event Delivery from Application Views

Application integration adapters generate events that are consumed by WebLogic Integration processes or by WebLogic Workshop Web services. Events are generated inside an EIS instance that is located outside of the WebLogic Integration cluster. Application integration is made aware of these events by event objects created by event connections within adapter instances. These adapter instances reside on individual managed servers within the cluster.

The behavior of event delivery from the point of origin in the EIS to the point where events are handed off to application integration is adapter-specific and not defined by application integration. For information about how adapter events are processed once the event is delivered to application integration, see "Events" on page 1-23.

In the case of a single managed server failure, any event in the process of being delivered to application integration on the failed node may be lost. Uninterrupted delivery of other events will continue if the following conditions exist:

- The adapter for the EIS supports multiple event connections within a cluster for a single application view or event type
- The event connection delivering the events has been deployed to more than one managed server in the cluster and at least one of those managed servers is still operational

Support for multiple event connections within a cluster for a single application view or event type depends on the design of the adapter. For example, the DBMS sample adapter included in the Adapter Development Kit does provide this support. If your adapter does not support multiple event generators, then deploying them in such a configuration could lead to multiple events being delivered to subscribers for a single EIS event. Consult your adapter vendor or adapter documentation to determine if the adapter you are using provides this support.

Deployment of the event connection to more than one managed server in the cluster depends on how you have configured the event connection and adapter instance. Using the WebLogic Integration Administration Console, WebLogic Integration administrators can target an event connection at zero or more managed servers in the cluster. If the adapter supports multiple event

generators in a cluster, it is best practice to deploy the event connection to all managed servers in the cluster.

For information on how to target event connections using the WebLogic Integration Administration Console, see Application Integration in *Managing WebLogic Integration Solutions*.

For information about how adapter events are processed by WebLogic Integration, see "Events" on page 1-23.

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. For more information, see Deploying Applications in the WebLogic Workshop Help.

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

- 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. For information on configuring these queues, see "To Manually Create Required Resources on the Production Server" in How do I: Deploy a WebLogic Workshop Application to a Production Server in the WebLogic Workshop Help.
- You need to create database tables for each conversational state element referred to in `wlw-manifest.xml`. For information on configuring these tables, see "To Manually Create Required Resources on the Production Server" in How do I: Deploy a WebLogic Workshop Application to a Production Server in the WebLogic Workshop Help.
- 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 WebLogic Server Applications*. For detailed information describing the deployment portion of the software lifecycle for WebLogic Platform applications, see *Deploying WebLogic Platform Applications*.

Deploying Adapters

An application integration adapter is typically composed of two components:

- A resource adapter deployed from a RAR file
- A design-time Web application deployed from a WAR file

The resource adapter (RAR) file should be deployed to the cluster. At a minimum, the resource adapter (RAR) file must be deployed to those managed servers where any application view using the adapter will be deployed. If an application view is deployed to a managed server that lacks the required adapter, the deployment of any adapter instances used by the application view and the application view deployment itself will fail.

The design-time Web application (WAR) file should not be targeted within the cluster. The design-time Web application file is used for development purposes only. WebLogic Integration production environments do not utilize this file.

For information about using the `weblogic.Deployer` command-line utility or the WebLogic Server Administration Console to deploy resource adapters to a running cluster, see Appendix A, "Deploying Resource Adapters." For more information about deploying adapters in the WebLogic Integration environment, see Deploying Adapters in *Developing Adapters*.

Deploying Event Generators

WebLogic Integration 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 *Managing WebLogic Integration Solutions*.

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 8.1 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 additonal 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:

If the JMS destination is a. . .	Target the. . .
Distributed destination	Cluster

Destination on a migratable server	Cluster **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.

For information about configuring an error queue, see "JMS Queue and Topic Destination Tasks" in JMS: Configuring in the WebLogic Server Administration Console Online Help.

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 "Starting the Administration Console" in Overview of WebLogic Server System Administration in *Configuring and Managing WebLogic Server*.

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 "Starting the Administration Console" in Overview of WebLogic Server System Administration in *Configuring and Managing WebLogic Server*.

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 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 WebLogic Integration in a clustered environment. To do this, you must configure a router (hardware or software), an administration server, and managed servers, and then deploy WebLogic Integration 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 WebLogic Integration in a clustered domain, complete the following steps:

- Step 1. Comply with Configuration Prerequisites
- Step 2. Prepare a WebLogic Integration Domain
- Step 3. Configure WebLogic Integration Security
- Step 4. Start and Monitor the Managed Servers in the Domain
- Step 5. Deploy a WebLogic Integration Application
- Step 6. Update Your Domain as Your Production Environment Changes

For information about deploying WebLogic Integration 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 a multicast address for each cluster

  Note: You are prompted to provide a multicast address when you create a WebLogic Integration domain using the Domain Configuration Wizard. (See "Step 2. Prepare a WebLogic Integration Domain" on page 4-5.)

  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.

- 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 WebLogic Integration domain using the Domain Configuration Wizard. (See "Step 2. Prepare a WebLogic Integration 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: `127.0.0.1:7003-7005`
ipAddress:portNumber+...+portNumber	When the port numbers are not sequential, for example: `127.0.0.1:7003+7006+7008`
ipAddress:portNumber,ipAddress:portNumber,...	Verbose, explicit specification, for example: `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 WebLogic Integration domain contains multiple clusters, specify the name of the WebLogic Integration 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 WebLogic Integration 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 WebLogic Integration 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 8.1 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 "Configuring WebLogic JMS Clustering" in Managing WebLogic JMS in *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 your Domain Configuration." For additional 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 a WebLogic Integration Domain

When preparing a WebLogic Integration domain, you must add a definition for each managed server to the domain configuration file (`config.xml`), assign all managed servers to a cluster, specify the WebLogic Integration components on the servers in your domain, and so on.

To prepare a WebLogic Integration environment in a clustered domain, complete the tasks described in the following sections:

- Creating a WebLogic Integration Domain Using the Configuration Wizard
- Creating the Database Tables

Creating a WebLogic Integration Domain Using the Configuration Wizard

You begin the definition of a clustered WebLogic Integration 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 Configurations Using the Configuration Wizard*.

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

1. From the Start Menu, choose **Programs—BEA WebLogic Platform 8.1—Configuration Wizard**.

   The Domain Configuration Wizard is launched. It prompts you for data with which to configure your domain.

2. Respond to the Domain Configuration Wizard prompts by providing the information described in the following table.

In this window . . .	Perform the following action . . .
Create or Extend a Configuration	Select Create a new WebLogic configuration.
Select a Configuration Template	Select the Basic WebLogic Integration Domain template
Choose Express or Custom Configuration	Select Custom.
Configure the Administration Server	Select or enter the administration server machine name or IP address for the Listen address. **Note:** When you configure the administration server, we recommend that you accept the default Server Name (`cgServer`), as prompted by the Domain Configuration Wizard.
Managed Servers, Clusters, and Machine Options	Do you want to distribute your WebLogic configuration across managed servers, clusters and physical machines? Select Yes.
Configure Managed Servers	Add as many managed servers as required. **Note:** If you need a http router for load balancing, add it here.
Configure Clusters	Add a cluster. **Note:** WebLogic Integration is intended to work with no more than one WebLogic Integration 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. **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.
Database (JDBC) Components	Do you want to define JDBC components, such as Connection Pools, Data Sources, and MultiPools? Select Yes.

Configure JDBC Connection Pools	Configure the JDBC Connection Pools for WebLogic Integration. For a description of the correct JDBC settings for an XA domain, see "How Do I: Create a WebLogic Integration Domain Using Oracle with an XA Driver?" in How Do I ...? in *Using the Configuration Wizard*. You must configure `cgPool` for WebLogic Integration to function. If the reporting data tables are referenced through a different data store than `cgPool`, then you must define `bpmArchPool` as a valid pool and configure it as the Reporting Data DataStore in the WebLogic Integration Administration Console. These configuration changes take effect on the next start of WebLogic Server. (For information on configuring the Reporting Data DataStore in the WebLogic Integration Administration Console, see "Configuring the Archive Data Store" in System Configuration in *Managing WebLogic Integration Solutions*.) You can choose additional Connection Pools for application use. If you are going to define a JDBC datastore, you must define at least one non-XA JDBC connection pool. JDBC datastores work only with non-XA JDBC connection pools. **Note:** WebLogic Integration may not be certified with all drivers. For a list of certified drivers, see "Supported Databases and Drivers" in Supported Configurations for WebLogic Platform in *Supported Configurations*.
Assign JDBC Connection Pools to MultiPools	Configure any Multipools needed by the application. Multipools are not required for the operation of WebLogic Integration itself.
Configure JDBC Data Sources	Accept the defaults for `cgDataSource` and `bpmArchDataSource`. Add any application-specific data sources. **Note:** No more than one non-XA datasource can be used in the same transaction. Data sources cannot share an XA JDBC connection pool.
Test JDBC Connection Pools and Setup JDBC Database	Optional test. For information about loading WebLogic Integration tables in the JDBC database, see "Creating the WebLogic Integration Tables" in Configuring a Production Database in *Managing WebLogic Integration Solutions*.

Messaging (JMS) Options	Do you want to define JMS components, such as Stores, Topics, and Queues? Select Yes. **Note:** Exercise caution in changing JMS option settings. Inappropriate settings may cause your cluster to function unpredictably. For more information about setting JMS options, see "Step 7: Configure the JMS Options" in "Tutorial: Creating a Custom Domain with Managed Servers, a Cluster and Application Services" in Tutorials: Using the Configuration Wizard in *Using the Configuration Wizard*.
Configure JMS Connection Factories	Accept the defaults for `cgQueue` and WLI-B2B System Topic Factory.
Configure JMS Destination Key(s)	Configure any keys needed for the application. None are needed for the operation of WebLogic Integration.
Configure JMS Template(s)	Configure any templates needed for the application. None are needed for the operation of WebLogic Integration.
Configure JMS File Stores	Accept the defaults for `rmfilestore` and add any file stores needed by the application.
Configure JMS JDBC Store	You should see one JMS Server for the administration server and one for each of the managed servers. Select the appropriate connection pool from the list, then add any JDBC stores needed by the application.
Configure JMS Servers	We recommend that you use one JMS Server per server rather than creating additional servers. WebLogic Integration System Queues can be mixed with Application Queues within the same JMS Server.
Assign JMS Servers to WebLogic Servers	Accept the defaults.
Configure JMS Topics	Accept the defaults, and add any user-defined topics.
Configure JMS Queues	Accept the defaults, and add any user-defined queues.
Configure JMS Distributed Topics	Accept the defaults, and add any user-defined distributed topics.
Configure JMS Distributed Queues	Accept the defaults, and add any user-defined distributed queues.
Assign JMS Distributed Destinations to Servers or Clusters	Accept the defaults.

Configure JMS Distributed Topic Members	Accept the defaults. There should be members for each configured managed server.
Configure JMS Distributed Queue Members	Accept the defaults.
Applications and Services Targeting Options	Do you want to target servers and clusters onto which Applications, JMS component services, JDBC component services, and other services are deployed? Select yes.
Target Applications to Servers or Clusters	Click Next. For information on the default targets for application components, see Appendix D, "WebLogic Integration Deployment Resources.".
Target Services to Servers or Clusters	Accept the defaults for non-XA data sources. Configure the appropriate target for XA data sources. For more information, see "Creating XA Domains Using Configuration Templates" in How Do I... in *Creating WebLogic Configurations Using the Configuration Wizard*.
Configure Administrative Username and Password	Select user names and passwords.
Configure Windows Options	Define operating system configuration information for Windows platforms. **Note:** Windows platform users only will encounter this page while using the Configuration Wizard. Configuring Windows options is not a requirement of a cluster domain configuration.

Configure Server Start Mode and Java SDK	Select Production Mode, and then select either the Sun SDK or JRockit SDK. **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. 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: • **32-bit Default** On either Windows or Linux IA32 machines, the default thread stack size values for native threads are: Win32: 64 kB Linux32: 128 kB • **64-bit Default** On either Windows or Linux IA64 machines, the default thread stack size values for native threads are: Win64: 320 kB Linux64: 1 MB
Create WebLogic Configuration	Select the name of your custom domain.

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

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 WebLogic Integration 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 WebLogic Integration uses.

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

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

Adding Proxy Server or Firewall Information to your 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.

Creating the Database Tables

You create database tables in a clustered environment following the same procedure as for a single-server deployment. For information about creating the WebLogic Integration database tables, see "Creating the Database Tables" on page 2-7.

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:

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 in Configuring, Starting, and Stopping Node Manager in *Configuring and Managing WebLogic Server*.

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 "Configure Startup Arguments for Managed Servers" in "Starting Managed Servers from the Administration Console" in Starting and Stopping Servers in *WebLogic Server Adminstration 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 *Using the Configuration Wizard*. For information about monitoring your WebLogic Integration domain, see "Run-Time Tuning Issues" in Performance Tips in the WebLogic Integration *Solutions Best Practices FAQ*.

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 WebLogic Integration 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 WebLogic Integration.

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

Step 5. Deploy a WebLogic Integration Application

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

If you did not configure all the queues necessary for your application while configuring your WebLogic Integration domain as described in "Creating a WebLogic Integration Domain Using the Configuration Wizard" on page 4-5, you can configure and target them 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 WebLogic Workshop development environment. You must create these queues and tables manually for production environments. For cluster deployments, these queues should be distributed destinations with physical members on each managed server.

For information on how to configure 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 on how to configure JMS resources using the WebLogic Server Administration Console, see "How Do I: Deploy WebLogic Workshop Web Services to a Production Server?" in How Do I... in WebLogic Workshop Help.

Note: If your WebLogic Integration 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 3-16.

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.

Note: You can update environment-specific information in your Application Views and adapter instances either before or after deploying your WebLogic Integration application:

- To update an unbooted domain before deployment, use the aiConfigurator as described in Appendix B, "Administering Environment-Specific Application Integration Information."
- To update after deployment (and redeploy, as necessary), use the WebLogic Integration Administration Console as described in Application Integration in *Managing WebLogic Integration Solutions.*

For examples of automation scripts that build, configure, and deploy WebLogic Integration applications outside of an interactive console environment, see the WebLogic Integration Solution Samples and the PO Sample that is available in the BEA dev2dev Code Library at the following URL:

`http://dev2dev.bea.com/code/wli.jsp`

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*.

Step 6. Update Your Domain as Your Production Environment Changes

Production environments change over time and as application use increases. This section describes how to update your domain in response to common production environment change scenarios:

- Adding a New Managed Server
- Changing an EIS Instance in a Cluster
- Installing a New Version of Your Application in a Cluster

For information about promoting WebLogic Platform applications from development environments to production environments, see "Steps to Promote WebLogic Platform Applications" in Overview of WebLogic Platform Development in *Deploying WebLogic Platform Applications*.

Adding a New Managed Server

As the use of an application grows, you may need to add new managed servers to a WebLogic Server cluster to provide extra capacity. For information about adding a new managed server to a cluster, see "Adding and Removing Servers in an Existing Domain" in Creating, Configuring, and Monitoring Servers in WebLogic Server Administration Console Online Help.

Once you have added the new managed server and started it within the cluster, you can begin to move processing responsibility onto that new server. To do this, complete the following procedure.

1. Decide which Application Views will be targeted at the new managed server.
2. Determine the list of adapters required to support the list of Application Views you arrived at in step 1.
3. Using the WebLogic Server Administration Console, target the RAR component of all the adapters identified in step 2 to the new managed server. Wait for the deployment to complete.

   For information about targeting a RAR component, see "Connector Component-->Configuration-->General" in the WebLogic Server Administration Console Online Help.

4. Using the WebLogic Server Administration Console, target the Application View EJBs to the new managed server. Wait for the deployment to complete.

   For information about targeting Application View EJBs, see "Deploying a New EJB Module" in EJB in the WebLogic Server Administration Console Online Help.

5. Using the WebLogic Integration Administration Console, verify that the Application Views you just targeted appear in the application view list as having a Deployed status.

   For information about verifying Application View status, see "Listing and Viewing Application Views" in Application Integration in the *Managing WebLogic Integration Solutions*.

6. If you have targeted the event connection for the Application Views to the cluster, event delivery will automatically begin from the new managed server. Otherwise, you (optionally) can specify the new managed server in the targets list for the event connection in the WebLogic Integration Administration Console on the Adapter Instance Event Connection page (**Application Integration→Adapter Instances→*Adapter_Instance_ID*→Edit Event Connections**).

   For more information about setting target lists for event connections, see "Changing Event Generation Targets" in Application Integration in the *Managing WebLogic Integration Solutions*.

After completing this procedure, Application View events should be coming from (if so configured) and Application View services should be invoked on the new managed server.

Changing an EIS Instance in a Cluster

The procedure for changing an EIS instance is the same in both single-server and cluster environments. For information about changing an EIS instance, see "Changing an EIS Instance" on page 2-10.

Installing a New Version of Your Application in a Cluster

The procedure for installing a new version of your application is the same in both single-server and cluster environments. For information about installing a new version of your application, see "Installing a New Version of Your Application" on page 2-10.

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.

The following sections describe clustering and high availability for a WebLogic Integration deployment:

- About WebLogic Integration High Availability
- WebLogic Integration Failure and Recovery

For additional recommendations and database-specific requirements for configuring high availability WebLogic Integration applications, see Maintaining Availability in the Deployment section of the WebLogic Integration documentation set located at the following URL:

`http://edocs.bea.com/wli/docs81/index.html`

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:

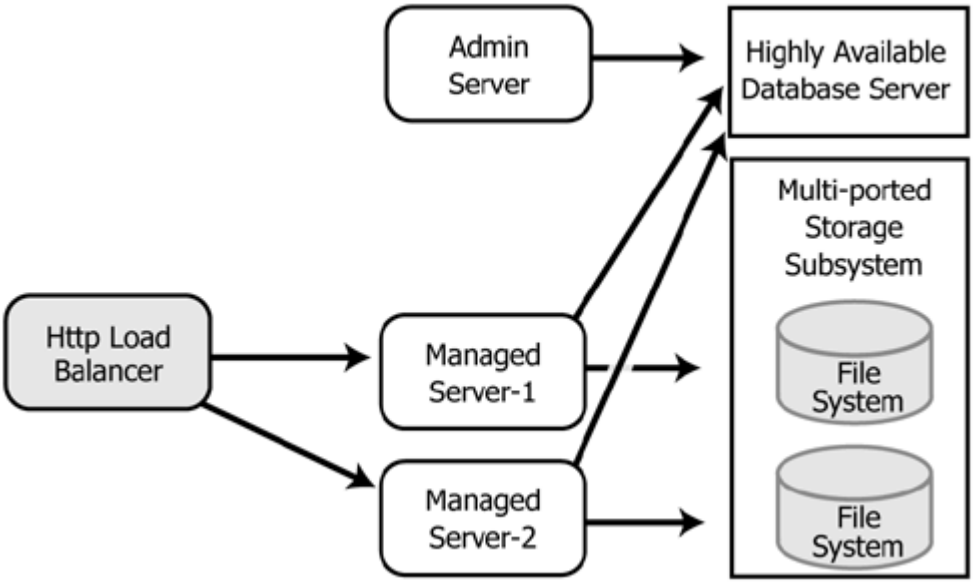
- An administration server.
- A set of managed servers in a cluster.
- An HTTP load balancer (router).
- Physically shared, highly-available disk subsystems for transaction recovery—Transaction logs from a failed server must be available to a managed server in order for migration to occur. A typical 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 "Configuring JDBC DataSources" in JDBC DataSources 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 "JMS Store Tasks" in JMS: Configuring in 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 in Configuring

and Managing WebLogic Server. For information about the steps to take to prepare for recovering a secure installation, see "Backing Up Configuration and Security Data" in Recovering Failed Servers in Configuring and Managing WebLogic Server.

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*
- Recovering Failed Servers in Configuring and Managing WebLogic Server

WebLogic Integration Failure and Recovery

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. The following sections discuss failure and recovery topics for specific WebLogic Integration functional areas:

- Trading Partner Integration
- Application Integration

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

Trading Partner Integration

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 *Managing WebLogic Integration Solutions*.

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 *Managing WebLogic Integration Solutions*.

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*.

Application Integration

WebLogic Integration provides you with great flexibility in managing application integration resources for high availability. The following sections describe how application integration resources behave in the case of software or hardware failures, and actions you can take to recover when failover is not automatic:

- Retargeting Services
- Retargeting Events
- EIS Instance Failover

Retargeting Services

In most cases, service invocations will continue uninterrupted because the application view containing the service is deployed to more than one managed server in the cluster. If this is not the case, use the WebLogic Server Administration Console to target the application view EJB and the adapter for the application view to a live managed server.

For information about using the WebLogic Server Administration Console to retarget services, see the following topics in the WebLogic Server Administration Console Online Help:

- "Setting an EJB Module's Target Server and/or Cluster" in Tasks.

- "Changing the Target Servers for a Deployment" in Tasks.

Retargeting Events

In the case of a single managed server failure, delivery of events targeted to other managed servers in the cluster continues. Uninterrupted delivery of events targeted to the failed managed server will continue if both of the following conditions exist:

- The event connection delivering the events has been deployed to more than one managed server in the cluster and at least one of those managed servers is still operational.
- The adapter for the EIS supports multiple event connections within a cluster for a single application view or event type.

If the event connection was targeted to the failed managed server only, use the WebLogic Integration Administration Console to target the event connection to a live managed server on which the application view is deployed. Retargeting the event connection will cause event delivery to resume on the targeted managed server.

Note: Changes to the event connection for a suspended application view do not apply to events already in the system. Only new events (those triggered after the change) are assigned to the new event connection target. Events already in the system are processed by the previous event connection target when the failed managed server returns to operation.

For information about using WebLogic Integration Administration Console to retarget event connections, see "Changing Event Generation Targets" in Application Integration in *Managing WebLogic Integration Solutions*.

EIS Instance Failover

When an EIS instance fails, all service invocations and event deliveries cease. Asynchronous service requests to the failed EIS instance will fail until the affected application views and adapter instances are placed in the Suspended state. If there is an operational instance of the EIS available, you can edit the affected application views and adapter instances to make use of the operational instance. (You use the WebLogic Integration Administration Console to perform these edits.) Otherwise, service invocations and event deliveries continue when you take application views and adapter instances out of the Suspended state.

The following sections describe suspending application views and adapters, resuming operation of application views and adapter instances, and retargeting application views and adapter instances to a different EIS instance:

- Suspending Application Views and Adapter Instances

- Resuming Operation
- Retargeting to an Operational EIS Instance

Suspending Application Views and Adapter Instances

In the case of an EIS failure, service requests and attempts at event delivery generate errors until the affected application views and adapter instances are suspended.

Note: Synchronous service requests to suspended application views and adapter instances fail. Asynchronous service requests to suspended application views and adapter instances are queued for processing when the suspended application views and adapter instances resume operation.

Application views and adapter instances can be suspended automatically or manually:

- If you have enabled the AutoSuspend option for application views and adapter instances (and the adapter supports auto-suspend functionality), the application views and adapter instances automatically go into the Suspended state in the case of an EIS instance failure.

  For information about using the WebLogic Integration Administration Console to enable the AutoSuspend option for application views, see "Viewing and Changing Application View Auto Suspend Settings" in Application Integration in *Managing WebLogic Integration Solutions*.

  For information about enabling the AutoSuspend option for adapter instances, see the JavaDoc for the `AdapterDeploymentMBean`.

  Note: The WebLogic Integration Administration Console allows you to enable the auto suspend option for all adapters, whether or not they provide auto-suspend functionality. Contact your adapter vendor to verify that your adapter supports auto suspend. (The DBMS sample adapter supports auto suspend.)

- If you have not enabled the AutoSuspend option or the adapter does not support auto-suspend functionality, you must first detect the EIS failure and then manually suspend the affected Application Views and adapter instances using the WebLogic Integration Administration Console.

  For information about using the WebLogic Integration Administration Console to suspend Application Views or adapter instances, see "Suspending or Resuming an Application View or Adapter Instance" in Application Integration in *Managing WebLogic Integration Solutions*.

You can detect an EIS instance failure using a monitoring tool for the EIS or by monitoring the error counts for the application view or adapter instance in the WebLogic Integration Administration Console. For information about monitoring errors using the WebLogic

Integration Administration Console, see Application Integration in *Managing WebLogic Integration Solutions*.

Resuming Operation

Once an EIS instance is again operational, you must remove the affected application views and adapter instances from their Suspended state in order for service requests and event delivery to resume.

For information about using the WebLogic Integration Administration Console to return application views or adapter instances to the Deployed state, see "Suspending or Resuming an Application View or Adapter Instance" in Application Integration in *Managing WebLogic Integration Solutions*.

Retargeting to an Operational EIS Instance

If you expect that an EIS instance failure will have an extended duration, you can point application views and adapter instances at an alternate, operational EIS instance. If an adapter instance already exists that points to an operational EIS instance, you can edit the Event Connection and Service Connection properties of any application view pointing to the failed EIS instance so that they are set to the adapter instance pointing to the operational EIS instance.

For information about using the WebLogic Integration Administration Console to change the adapter for an application view, see "Changing Event Connections for an Application View" and "Changing Service Connections for an Application View" in Application Integration in *Managing WebLogic Integration Solutions*.

You can also edit the Event Connection and Service Connection properties that point to the old EIS instance, and give the Event and Service Connections new property values to point them to the new, operational EIS instance.

For more information on changing Event and Service Connection properties, see "Viewing and Changing Event Connection Properties" and "Viewing and Changing Service Connection Properties" in Application Integration in *Managing WebLogic Integration Solutions*.

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 Introducing WebLogic Platform 8.1 Security in *Introducing Security*.

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 WebLogic Integration

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 "Default Groups, Roles, and Security Policies" in User Management in *Managing WebLogic Integration Solutions*.
- PasswordStore, which is described in "WebLogic Integration PasswordStore for Encrypted Passwords" on page 6-2.
- Default identity and trust keystores, which are described in "Keystore for Private Keys and Certificates" on page 6-3.
- B2BDefaultWebAppApplication, on which you can configure policies to authorize access to Trading Partner Integration.

For information about using the Configuration Wizard, see *Creating WebLogic Configurations 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 *Managing WebLogic Integration Solutions*:

- 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: • client • server • signature • encryption 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 *Managing WebLogic Security*.

2. Configure the keystores using the WebLogic Server Administration Console according to the instructions in "Configuring KeyStores" in Configuring SSL in *Managing WebLogic Security*.
3. Add trading partner certificates to the identity keystore. For more information, see "Step 3: Configure Application Integration Security" on page 6-12.
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 *Managing WebLogic Integration Solutions*.

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 WebLogic Integration

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 *Managing WebLogic Integration Solutions*.

- *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 *Managing WebLogic Integration Solutions*.

- *Application Views*

  Policies define the roles required to execute services and subscribe for events on an application view. For more information, see "Managing Security" in Application Integration in *Managing WebLogic Integration Solutions*.

- *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 *Managing WebLogic Security*.

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" on page 6-11
- "Step 2: Configure WebLogic Server Security" on page 6-11
- "Step 3: Configure Application Integration Security" on page 6-12
- "Step 4: Configure Web Application and Web Service Security-Related Deployment Descriptors" on page 6-13
- "Step 5: Configure Security Policies and Manage Users" on page 6-14
- "Step 6: Configure Worklist Security" on page 6-19
- "Step 7: Configure Trading Partner Integration Security" on page 6-19

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 your Domain Configuration" on page 4-11.

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 *Managing WebLogic Security*.

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 Application Integration Security

WebLogic Integration provides the following security mechanisms for those parts of an integration solution that are created and maintained with application integration functionality:

- To connect to an Enterprise Information System (EIS), an application might need to provide certain credentials, such as a login name and password.

  For more information, see "Scenario 1: Connecting Using Specific Credentials" in Using Application Views by Writing Custom Code in *Using the Application Integration Design Console*.

- When deploying an application view, you can configure security settings to grant or revoke read and write access to the application view by a WebLogic Server user or group.

  For more information, see "Steps for Defining an Application View" in Defining an Application View in *Using the Application Integration Design Console*.

Step 4: Configure Web Application and Web Service Security-Related Deployment Descriptors

Using WebLogic Workshop, a developer can edit Web application settings and Web 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 the following:

  – "Web Application Security-Related Deployment Descriptors" in Securing Web Applications in *Programming WebLogic Security*

  – "Defining a Protected Web Resource" in Security in the WebLogic Workshop Help.

- In `wlw-config.xml`, the web service exposure protocol should be set to HTTPS.

  For information about configuring security options in `wlw-config.xml`, see "wlw-config.xml Configuration Files" in the WebLogic Workshop Help.

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

Warning: Redeploying an application from WebLogic 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 "How Do I: Deploy WebLogic Workshop Web Services to a Production Server" in the WebLogic Workshop Help.

Step 5: 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 "Starting the Administration Console" in Introducing the WebLogic Integration Administration 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 *Managing WebLogic Integration Solutions*.

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 *Managing WebLogic Integration Solutions*.
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 

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 *Managing WebLogic Integration Solutions*.

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 

in the module navigation bar to return to the home page.

Configuring Security Policies for Application Views

1. On the home page of the WebLogic Integration Administration Console, click **Application Integration**, and then choose the **Application Views** tab to list the AppViewID for each application view in the WebLogic Integration application.

2. Click an **AppViewID** to access the **Application View Details** page.

   From the **Application View Details** page, you can configure the following security settings:

   - Roles authorized to execute services and subscribe for events for the application view.

     For the procedure to configure these security policies, see "Updating Security Policies" in Application Integration in *Managing WebLogic Integration Solutions*.

   - Container-managed sign-on.

     Enabling container-managed sign-on for an Application View allows Application Views to utilize any principal map that has been configured for the service connection. If container-managed sign-on is enabled for an Application View and the service connection it uses has been configured with a principal map, then at runtime the services invoked on the Application View will execute within the EIS instance with the identity of the EIS principal that maps to the WebLogic Server principal in effect when the service was invoked. If container-managed sign-on is disabled or no principal map exists on the service connection, then the authentication properties in the service connection itself (if any) are used to connect to the EIS instance

For the procedure to configure this setting, see "Enabling or Disabling Container-Managed Sign-On" in Application Integration in *Managing WebLogic Integration Solutions.*

3. Click **View All** to return to the **Application View Summary** page, and repeat step 2 for each application view.

4. After configuring the security settings for each application view, click 

in the module navigation bar to return to the home page.

Configuring Security Policies for Adapter Instances

1. On the home page of the WebLogic Integration Administration Console, click **Application Integration**, and then choose the **Adapter Instances** tab.

   The **Adapter Instance Summary** page lists the ID for each adapter instance in the WebLogic Integration application.

2. Click an ID to access the **Adapter Instance Details** page, and then click **Select Service Connection** to display the **Adapter Instance Service Connection** page.

   The **Adapter Instance Service Connection** page lists the name of each service connection for the adapter instance.

3. Click the name of a service connection to access the **Adapter Instance Service Connection Details** page.

   From the **Adapter Instance Service Connection Details** page, you can configure the following security settings:

   – Roles authorized to obtain service connections from the connection factory

     For the procedure to configure these security policies, see "Updating Security Policies" in Application Integration in *Managing WebLogic Integration Solutions*.

   – WebLogic Server to EIS principal mappings

     For the procedure to configure this map, see "Viewing and Changing WebLogic Server to EIS Principal Mappings" in Application Integration in *Managing WebLogic Integration Solutions*.

4. Repeat step 3 for each service connection.

5. After configuring the security settings for each service connection, click **View All** to return to the **Adapter Instance Summary** page, and repeat steps 2, 3, and 4 for each adapter instance.

6. After configuring the security settings for each channel, click 

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 6: Configure Worklist Security

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

- IntegrationUser—All users performing worklist tasks must be assigned to the IntegrationUser role. Users automatically inherit the IntegrationUser role through their default membership in the IntegrationUsers group.
- TaskCreationRole—Users and groups creating worklist tasks must be manually assigned to the TaskCreationRole.

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 6-18.) 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. For more information about configuring worklist security, see User Management and Worklist Administration in *Managing WebLogic Integration Solutions*.

Step 7: 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

Deploying Resource Adapters

This section describes how to deploy resource adapters after you start the servers in your cluster. For information about how to set up and start your clustered deployment, and which adapters are deployed by default in your WebLogic Integration domains, see Chapter 4, "Configuring a Clustered Deployment."

After you start the servers in your cluster, you can deploy resource adapters by using one of the following methods:

- Using the weblogic.Deployer Command-Line Utility
- Using the WebLogic Server Administration Console

Using the weblogic.Deployer Command-Line Utility

The weblogic.Deployer utility is a Java-based deployment tool that provides a command-line interface to the WebLogic Server deployment API. For information, see "weblogic.Deployer Utility" in Deployment Tools Reference in *Deploying WebLogic Server Applications*.

Deploying the Sample DBMS Adapter

The following example demonstrates how to deploy the sample DBMS adapter, which you received with your WebLogic Integration software, into a cluster named MyCluster. The cluster contains two managed servers: MyServer1 and MyServer2. The following table describes the cluster configuration.

Server Name	Server Type	Listen Address:Port
MyAdmin	Administration Server	127.0.0.5:7005
MyServer1	Managed Server	127.0.0.1:7001
MyServer2	Managed Server	127.0.0.1:7002

Use the following command to deploy the DBMS adapter in this example cluster.

Note: The following code listing represents a single command. It is shown here on multiple lines for the sake of readability. On your command line, however, it must be entered as one physical line.

Listing A-1 weblogic.Deployer Command Line to Deploy the DBMS Adapter

```
java -classpath WL_HOME\lib\weblogic.jar weblogic.Deployer
-adminurl t3://127.0.0.5:7005 -user username -password password
-upload -stage
-source WL_HOME\adapters\dbms\lib\BEA_WLS_DBMS_ADK.ear
-name BEA_WLS_DBMS_ADK
-targets BEA_WLS_DBMS_ADK.rar@MyCluster
-activate
```

In the preceding command line:

- `-adminurl`—Specifies the URL for the administration server in the cluster.
- `-user`—Specifies the username used for authentication by the administration server.
- `-password`—Specifies the username used for authentication by the administration server.
- `-upload`—Uploads the EAR file to the administration server. You can omit this option when you run the `weblogic.Deployer` utility on the administration server. However, it is required when you are not running the `weblogic.Deployer` utility on the administration server.

- `-stage`—Instructs the WebLogic Server deployment facility to stage the EAR file to all managed servers prior to activation.
- `-source`—Specifies the location of the EAR file for the resource adapter. (*`WL_HOME`* represents the directory in which you installed WebLogic Integration, for example `C:\bea\weblogic81\integration`.)
- `-name`—Specifies the name of the enterprise application for the resource adapter, which should be the same as the logical name for the adapter. This is a unique identifier for a resource adapter.
- `-targets`—Specifies the subcomponents contained in the previously specified EAR file for the adapter.

  This is a comma-separated list of subcomponents. (There are no spaces between the items in the list.) This sample command specifies that the RAR is deployed to the cluster.

  Note: The RAR component from the EAR—not the design-time Web application—is the target for deployment.

  For details about valid target components, see Appendix D, "WebLogic Integration Deployment Resources.".
- `-activate`—Activates the application in the domain.

Using the WebLogic Server Administration Console

1. 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.

2. In the WebLogic Server Administration Console navigation tree, select (***Domain_Name*→Deployments→Applications**) the Applications node in the domain in which you want to deploy an adapter.

3. Click **Configure a New Application**.

   The WebLogic Server wizard is displayed in the main console window. It guides you through the process of configuring and deploying your adapter.

4. Locate the EAR, WAR, JAR, or RAR file you would like to configure for use with WebLogic Server. For example, to deploy the sample DBMS adapter, which you received with your WebLogic Integration software, select the `BEA_WLS_DBMS_ADK.ear` file in the following directory:

   `WL_HOME``\adapters\dbms\lib\BEA_WLS_DBMS_ADK.ear`

   In the preceding line, *`WL_HOME`* represents the directory in which you installed WebLogic Integration, for example, `C:\bea\weblogic81\integration`.

   Note: When you configure an exploded application or component directory, WebLogic Server deploys all components it finds in and below the specified directory.

5. Complete the configuration and deployment by responding to the prompts in the wizard. For example, you must specify the targets and the staging mode. For more information, see the `-targets` and `-stage` options in "Using the weblogic.Deployer Command-Line Utility" on page A-1.

For information about using the WebLogic Server Administration Console to deploy applications, see "Configuring and Deploying a New Enterprise Application or Web Service" in Enterprise Applications in the *WebLogic Server Adminstration Console Online Help*.

APPENDIX B

Administering Environment-Specific Application Integration Information

This section describes how to use the `aiConfigurator` utility to modify environment-specific information for application view, adapter, and connection factory descriptors.

Note: The `aiConfigurator` utility should only be run on a domain which has not been started. Do not use this utility on a running domain server. Using the utility on a running domain server can result in unpredictable behavior.

aiConfigurator Utility and Examples

Application views, by using environment variables, can have environment-specific information parameterized and isolated from business-oriented information. With this parameterization comes the need to modify the parameter values to reflect the needs of new environments. Adapter instances and connection factories can also include environment-specific information.

The `aiConfigurator` utility (based on Java class `com.bea.wlai.management.util.Configurator`) allows an administrator to modify environment-specific information across application view, adapter, and connection factory descriptors. This allows an administrator to preconfigure application integration resources to deploy correctly in a new target environment. Further tuning of these resources can then be performed using the WebLogic Integration Administration Console. The `aiConfigurator` utility is located in:

`WL_HOME``/integration/bin/aiConfigurator.cmd` (or `.sh`)

This utility updates the WebLogic Integration configuration persistent store, and optionally publishes application view EJB contents, to reflect the needs of the new environment. At runtime,

the newly tailored information is fetched from the persistent store and applied to the in-memory state of the application view, adapter instance, or connection factory.

"Switching Database Type/Instance for DBMS Sample Adapter" on page B-5 provides an example of the use of the `aiConfigurator` utility, and shows how to change database types/instances when using the DBMS sample adapter. For more information on the DBMS sample adapters, see *Developing Adapters*.

aiConfigurator Usage

The `aiConfigurator` utility is ordinarily used to override application view environment variables, and adapter or connection factory settings at runtime, leaving the original descriptors intact. This utility also allows for the default values of application view environment variables to be replaced with the values specified, and adapter instance or connection factory settings to be persisted into the original descriptors. The latter capability is useful for samples that the administrator expects users to edit later.

The usage of the `aiConfigurator` utility is as follows:

```
aiConfigurator -appName app_name -appFile app_file
-domainRootDir domain_root_dir [-updateDesignTime]
```

Plus one of the following groups of arguments:

```
[ -configAppView
  -appViewName app_view_name
  [ -dump |
    < -vars vars|properties_file
      -var name=value
      -eventAdapterName qualified_name
      -serviceAdapterName qualified_name
      -serviceFactoryName name
      -autoSuspendEnabled true|false
      -autoSuspendTimeout integer_seconds
      -suspendedRequestRetryInterval integer_seconds
      -suspendedEventRetryInterval integer_seconds
    >

    ]
  ]

[ -configAdapter
```

```
  -appViewName app_view_name
  -adapterName adapter_instance_name
  [ -dump |
    < -props props|properties_file
      -prop name=value
      -inboundMessagingTargets comma-separated_server_names
      -autoSuspendEnabled true|false
      -autoSuspendTimeout integer_seconds
    >
    ]
  ]

[ -configFactory
  -appViewName app_view_name
  -adapterName adapter_instance_name
  -factoryName connection_factory_name
  [ -dump |
    < -props props|properties_file
      -prop name=value
      -minPoolSize integer
      -maxPoolSize integer
    >
    ]
  ]
```

where:

- `appName` is the name of the application to configure
- `appFile` is the directory or EAR archive containing the application
- `domainRootDir` is the root directory of the domain the application is to be deployed into
- `updateDesignTime` is used to force the changed values back into the design-time artifacts and the WebLogic Integration persistent configuration store. If this argument is not used, all changes are saved adjacent to the unchanged design-time artifacts in the WebLogic Integration persistent configuration store. For application views, the only changes that are persisted back to the design-time descriptor are variable values. All other settings are persisted only to the WebLogic Integration persistent configuration store.
- `vars` and `props` are in form *`name`*=*`value`*

For application views, you can update the following types of information:

Note: The only Application View information that can be updated for the design-time descriptor using the `-updateDesignTime` argument are the environment variables.

- Environment variables—Supply name and value pairs with `-var` arguments or as a properties file given with a `-vars` argument.
- Event adapter—Supply the event adapter name with the `-eventAdapterName` argument.

  Note: Use this option with caution. The name you specify should be fully qualified and must represent an adapter instance you know will already be deployed in the integration server (for example, if another application view uses the adapter instance, and you have guaranteed the other application view will deploy before the current application view based on `DeploymentOrder` and other settings).
- Service adapter—Supply the service adapter name with the `-serviceAdapterName` argument.

  Note: Use this option with caution. The name you specify should be fully qualified and must represent an adapter instance you know will already be deployed in the integration server (for example, if another application view uses the adapter instance, and you have guaranteed the other application view will deploy before the current application view based on `DeploymentOrder` and other settings).
- Service connection factory—Supply the connection factory name with the `-serviceFactoryName` argument.

  Note: The name you give must be the name of a connection factory within the adapter instance used for services on this application view (`-serviceAdapterName` argument).
- Enable or disable the auto-suspend feature—You can enable or disable the auto-suspend feature by setting the `-autoSuspendEnabled` argument to `true` or `false`.
- Auto-suspend timeout—Supply the number of seconds for the auto-suspend timeout using the `-autoSuspendTimeout` argument.
- Suspended-request retry interval—Supply the suspended-request retry interval (in seconds) with the `-suspendedRequestRetryInterval` argument.
- Suspended-event retry interval—Supply the suspended-event retry interval (in seconds) with the `-suspendedEventRetryInterval` argument.

  Note: Currently, the suspended event retry interval value is not used by the `aiConfigurator` utility, and the `AppViewDeploymentMBean` does not define a

`setSuspendedEventRetryInterval()` method. The suspended request retry interval value is used for both asynchronous service request and event retry intervals.

For adapter instances you can update the following information:

Note: The only adapter instance information that can be updated for the design-time descriptor using the `-updateDesignTime` argument are the event generation properties.

- Event generation properties—Supply name and value pairs or a properties file with a `-props` argument.
- Inbound messaging target—Supply a comma-separated list of machines to which inbound messaging is targeted using the `-inboundMessagingTargets` argument.
- Enable or disable the auto suspend feature—You can enable or disable the auto-suspend feature by setting the `-autoSuspendEnabled` argument to `true` or `false`.
- Auto suspend timeout—Supply the number of seconds for the auto-suspend timeout using the `-autoSuspendTimeout` argument.

For connection factories you can update the following information:

Note: The only connection factory information that can be updated for the design-time descriptor using the `-updateDesignTime` argument are the service invocation properties.

- Service invocation properties—Supply name and value pairs or a properties file with `-prop` arguments.
- Pool size—Supply the minimum and maximum pool size for connections with the `-minPoolSize` and `-maxPoolSize` arguments.

You can determine the current settings for each application integration artifact type by passing the `-dump` argument after the `-config*` argument. This is useful if the environment-specific configuration occurs in steps, or changes over time.

Switching Database Type/Instance for DBMS Sample Adapter

As an example of how to use the `aiConfigurator` utility, this section discusses the WebLogic Integration sample application and how to configure the samples in this application to execute on an Oracle database as opposed to the PointBase database.

To configure the samples, run the `aiConfigurator` utility against each application integration artifact that needs to be reconfigured. The current `sampleApp/ApplicationIntegration` directory contains two application views (`FunctionDemo.CustomerMgmt` and

InsertBasedEvents). These application views each use a single adapter instance, and a single connection factory within this adapter instance. The aiConfigurator utility is run against each of these artifacts in turn.

For each application view, update its variable set to reflect the new Oracle environment. For the DBMS sample adapter, this means setting the catalog and schema qualifiers for the tables that are used in events and services.

For the application integration samples, we've defined three variables:

- myCatalog—The catalog containing the schema that contains the CUSTOMER_TABLE used for the insert and update events in CustomerMgmt and InsertBasedEvents, respectively.
- mySchema—The schema containing the CUSTOMER_TABLE.
- myTableQualifiers—A variable combining catalog and schema to form a prefix for table names in SQL queries.

For the adapter instances, the event generation properties need to be updated to reflect the correct catalog and schema for the event staging tables.

For the connection factories, the DB type, JDBC driver URL, and other properties specific to the original PointBase environment need to be updated. These properties are changed using the switchDB script for your platform. This script uses the -updateDesignTime argument of the aiConfigurator utility to force updates back into the design-time artifacts, thus allowing edits of these artifacts from within the Oracle design-time environment.

In summary, to switch databases used by the application integration samples and the samples domain, do the following:

1. Change the JDBCConnectionPool elements in the domain's config.xml file to point to the Oracle instance. This involves specifying the JDBC driver class name and JDBC URL, and setting the pool properties user/password for the new database instance. See your WebLogic Server documentation for details.
2. Run the switchDb script specific to your operating system within the *WL_HOME*/samples/integration/sampleApp/ApplicationIntegration directory. This updates the application view, adapter, and connection factory descriptors contained in the sampleApp to reflect the new database type/instance. Note that usage for the switchDb utility is:

   Usage: switchDb (*db_type*) (*db_server*) (*db_name*) (*db_user*) (*db_password*)

APPENDIX C

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 WebLogic Workshop run-time engine. For information about `jws-config.properties`, see jws-config.properties Configuration File in the WebLogic Workshop Help.

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

Table C-1 Contents of wli-config.properties file

Property	Description	Default value
`weblogic.wli.DocumentMaxInlineSize`	Minimum size in bytes for documents stored in the SQL Document Store	`524288`
`weblogic.wli.DocumentMaxInMemorySize`	Maximum size in bytes of document buffered before writing to the SQL Document Store	`524288`
`weblogic.wli.ProcessDocumentCleanupIntervalMinutes`	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-1 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. **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.TrackingPurgeTxnTimeMillis`	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. **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 D

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 "Configuring JMS Migratable Targets" and "Configuring WebLogic JMS Clustering" in Managing WebLogic JMS in *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
  - `wlai` corresponds to application 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	Administration Console Navigation
Admin Server	WLI Console `wliconsole.war`	Console
	WLI AI RAR Upload `wli-ejbs.ear/wlai-rarupload-ejb.jar`	*Domain*→ Deployments→ Applications→ WLI System EJBs
	WLI Calendar Persistence `wli-ejbs.ear/calendar/generic`	*Domain*→ Deployments→ Applications→ WLI System EJBs
	`wli.internal.b2b.events.topic`[1] (Topic for cluster events)	*Domain*→ Services→ JMS→ Servers→ cgJMSServer→ Destinations
	`wli.internal.configfile.request.queue`[1] (Return queue to configuration manager)	
	`wli.internal.configfile.update.topic`[1] (Update topic for configuration manager)	
	B2BDefaultWebAppApplication `applications/B2BDefaultWeb`	*Domain*→ Deployments→ Web Application Modules
	`cgQueue`[1] (WLI/WLW Connection factory)	*Domain*→ Services→ JMS→ Connection Factories

Table D-1 WebLogic Integration Deployment Resources (Continued)

Resource Target	Resource	Administration Console Navigation
Admin Server (continued)	WLI Post-Activation Startup Class `com.bea.wli.init.BPMStartupAfterActivation`	*Domain*→ Deployments→ Startup & Shutdown
	WLI Shutdown Class `com.bea.wli.init.BPMShutdown`	
	WLI Startup Class `com.bea.wli.init.BPMStartup`	
	WLI-B2B System Topic Factory (WLI-B2B Connection factory)	*Domain*→ Services→ JMS→ Connection Factories
	`bpmArchPool`[1] (WLI JDBC Pool for Archiving Database Tables)	*Domain*→ Services→ JDBC→ Connection Pools
	`cgPool`[1] (WLI/WLW JDBC Pool)	
Cluster	BEA_WLS_DBMS_ADK `BEA_WLS_DBMS_ADK.ear/BEA_WLS_DBMS_ADK.rar` (BEA_WLS_DBMS_ADK Connector)	*Domain*→ Deployments→ Applications→ BEA_WLS_DBMS_ADK
	BEA_WLS_DBMS_ADK_LOCALTX `BEA_WLS_DBMS_ADK.ear/BEA_WLS_DBMS_ADK_LOCALTX.rar` (BEA_WLS_DBMS_ADK Connector)	
	BEA_WLS_DBMS_ADK_Web `BEA_WLS_DBMS_ADK.ear/BEA_WLS_DBMS_ADK_Web` (BEA_WLS_DBMS_ADK Web Application)	

Table D-1 WebLogic Integration Deployment Resources (Continued)

Resource Target	Resource	Administration Console Navigation
Cluster (continued)	WLI Admin `wli-ejbs.ear/wliadmin`	*Domain*→ Deployments→ Applications→ WLI System EJBs
	WLI Admin Helper `wli-ejbs.ear/adminhelper`	
	WLI AI Message Processors `wli-ejbs.ear/wlai-processors-ejb.jar`	
	WLI Calendar Persistence `wli-ejbs.ear/calendar/generic`	
	WLI ebXML `wli-ejbs.ear/ebxml`	
	WLI Message Tracking `wli-ejbs.ear/message-tracking`	
	WLI Process Proxy Dispatcher `wli-ejbs.ear/proxydispatcher`	
	WLI Process Tracking `wli-ejbs.ear/tracking`	
	WLI RosettaNet `wli-ejbs.ear/rosettanet`	
	WLI Sync2Async Transport Servlet `wli.internal.SyncAsyncResponseListener`	
	WLI Sync2Async Response Listener `wli.internal.SyncAsyncTransportServlet`	
	WLI Worklist Persistence `wli-ejbs.ear/worklist/persistence/generic`	

Table D-1 WebLogic Integration Deployment Resources (Continued)

Resource Target	Resource	Administration Console Navigation
Cluster (continued)	WLI Worklist Selection `wli-ejbs.ear/worklist/selection`	*Domain*→ Deployments→ Applications→ WLI System EJBs
	WLI-B2B HTTP Transport `wli-ejbs.ear/b2btransport-webapp`	
	`.workshop/worklist/EJB/GenericStateless` `worklistApp.ear/.workshop/worklist/EJB/GenericStateless`[1] (Worklist user interface)	*Domain*→ Deployments→ Applications→ WLI Worklist
	`.workshop/worklist/EJB/ProjectBeans` `worklistApp.ear/.workshop/worklist/EJB/ProjectBeans`[1] (Worklist user interface)	
	`worklist` `worklistApp.ear/worklist`	
	wlai `wlai-designtime.ear/wlai.war`	*Domain*→ Deployments→ Applications→ WLI-AI Design-time
	WLI-AI Manager EJBs `wlai-designtime.ear/wlai-manager-ejb.jar`	
	JWSQueueTransport `QueueTransportEJB.jar`	*Domain*→ Deployments→ EJB Modules
	WLI Post-Activation Startup Class `com.bea.wli.init.BPMStartupAfterActivation`	*Domain*→ Deployments→ Startup & Shutdown
	WLI Shutdown Class `com.bea.wli.init.BPMShutdown`	

Table D-1 WebLogic Integration Deployment Resources (Continued)

Resource Target	Resource	Administration Console Navigation
Cluster (continued)	WLI Startup Class `com.bea.wli.init.BPMStartup`	*Domain*→ Deployments→ Startup & Shutdown
	`cgQueue`[1] (WLI/WLW Connection Factory)	*Domain*→ Services→ JMS→ Connection Factories
	`wli.internal.egrdbms.XAQueueConnectionFactory`[1] (RDBMS Event Generator Connection Factory)	
	`cgPool`[1] (WLI/WLW JDBC Pool)	*Domain*→ Services→ JDBC→ Connection Pools
	`bpmArchPool`[1] (WLI JDBC Pool for Archiving Database Tables)	
	`bpmArchDataSource`[1] (WLI Data Source for Archiving)	*Domain*→ Services→ JDBC→ Data Sources
	`cgDataSource`[1] (WLI/WLW Data Source)	
	`WLAI_DataSource`[1] (Data source for WLAI Sample Applications)	
	B2BDefaultWebAppApplication `applications/B2BDefaultWebAppApplication`	*Domain*→ Deployments→ Web Application Modules
Each managed server	`cgJMSServer`[1, 2] (WLI/WLW JMS Server)	*Domain*→ Services→ JMS→Servers

Table D-1 WebLogic Integration Deployment Resources (Continued)

Resource Target	Resource	Administration Console Navigation
Each managed server	`wli.internal.ai.event_suspend`[1] (Application Integration suspended event queue)	*Domain*→ Services→ JMS→ Servers→ cgJMSServer→ Destinations
Distributed Destination	`wli.internal.msgtracking.queue`[1] (Message tracking JMS queue)	*Domain*→ Services→ JMS→ Servers→ cgJMSServer→ Destinations
	`wli.internal.ai.async.request`[1] (Application Integration asynchronous request queue)	
	`wli.internal.ai.async.response`[1] (Application Integration asynchronous response queue)	
	`wli.internal.instance.info.buffer`[1] (Application Integration instance information queue)	
	`wli.internal.instance.info.buffer_error`[1] (Errors associated with Application Integration suspended event queue)	
	`wli.b2b.mt.event.stream`[1] (Business process message tracking event queue)	
	`wli.b2b.mt.event.stream_error`[1] (Errors associated with business process message tracking event queue)	
	`wli.internal.ai.async.request`[1] (Application Integration asynchronous request queue)	

Table D-1 WebLogic Integration Deployment Resources (Continued)

Resource Target	Resource	Administration Console Navigation
Distributed Destination (continued)	`wli.internal.ai.event`[1] (Application Integration event topic)	*Domain*→ Services→ JMS→ Servers→ cgJMSServer→ Destinations
	`wli.internal.ai.event_suspend`[1] (Application Integration suspended event queue)	
	`wli.internal.b2b.ebxmlencoder.queue`[1] (ebXML outbound JMS queue)	
	`wli.internal.b2b.rosettanetencoder.queue`[1] (RosettaNet outbound JMS queue)	
	`wli.internal.sync2Async.soapResponse`[1] (Asynchronous business process to synchronous client inbound queue)	
	`wli.internal.tracking.buffer`[1] (Queue for process tracking)	
	`wli.internal.tracking.buffer_error`[1] (Errors associated with queue for worklist timers)	
	`wli.internal.worklist.timer.queue`[1] (Queue for worklist timers)	
	`wli.process.event.stream`[1] (Queue for process events)	
	`wli.process.event.stream_error`[1] (Errors associated with queue for process events)	

Table D-1 WebLogic Integration Deployment Resources (Continued)

Resource Target	Resource	Administration Console Navigation
Migratable Server	`wli.internal.scheduling.queue`[1] (Timer queue for process archiving)	*Domain*→ Services→ JMS→ Servers→ cgJMSServer→ Destinations
	`wli.internal.scheduling.queue_error`[1] (Error queue for process archiving)	
	`wli.internal.SQLStore.cleanup.documents`[1] (Timer queue for doc store cleanup)	
	`wli.internal.egfile.queue`[1] (Timer queue for file event generator)	
	`wli.internal.egmail.queue`[1] (Timer queue for email event generator)	
	`wli.internal.egmq.queue`[1] (Timer queue for MQSeries event generator)	
	`wli.internal.egtimer.queue`[1] (Timer queue for timer event generator)	

1. WebLogic Server Administration Console displays the package or service name.
2. The Configuration Wizard labels JMS Servers uniquely for each managed server with names like `cgJMSServer_auto_1`.

Index

E

K

L

M

N

O

P

Q

R

S

T

W

X