BEA WebLogic Integration™

Deploying BEA WebLogic Integration Solutions

Release 7.0
Document Date: June 2002

Copyright

Restricted Rights Legend

Trademarks or Service Marks

Deploying BEA WebLogic Integration Solutions

Part Number	Date	Software Version
N/A	June 2002	7.0

Contents

2. Understanding WebLogic Integration Clusters

3. Configuring a Clustered Deployment

4. Understanding WebLogic Integration High Availability

5. Using WebLogic Integration Security

6. Tuning Performance

About This Document

This document describes how to deploy an integration solution using BEA WebLogic Integration in a production environment. Specifically, it describes how to deploy an integration solution that meets goals for high availability, performance, scalability, and security. It defines key deployment concepts, explains how to deploy integration solutions on a WebLogic Integration cluster, provides an overview of WebLogic Integration security, and describes how to tune performance in a production environment.

Overview Documents for WebLogic Integration

This document is one in a series of four documents that provide an overview of WebLogic Integration, and that explain how the functionality provided by WebLogic Integration is used at various stages in the design, development, and deployment of integrated solutions. Readers should start with these documents to gain a comprehensive understanding of the functionality provided by WebLogic Integration. The other documents in the series are:

- *Introducing BEA WebLogic Integration*—Provides an overview of WebLogic Integration. It outlines the integration problems faced by e-businesses today, as they try to conduct business with collections of fragmented, heterogeneous systems. It also describes the application integration, B2B integration, business process management, and data integration functionality provided by WebLogic Integration to solve e-business integration problems.
- *Learning to Use BEA WebLogic Integration*—Describes a sample integrated application. The sample application deploys a supply-chain hub, which connects

with business partners, automates a number of business processes, and integrates back-end enterprise information systems. Readers learn how to set up and run the sample application, and understand how the integrated solution is architected and developed using WebLogic Integration.

- *Designing BEA WebLogic Integration Solutions*—Describes how to design an integration solution in the BEA WebLogic Integration environment. It defines key design concepts, provides a roadmap for determining integration requirements, based on a comprehensive analysis of business and technical requirements, and describes how to design an integration architecture that meets design goals for high availability, scalability, and performance.

These and other WebLogic Integration documents are available at the following URL:

`http://edocs.bea.com/wli/docs70/index.html`

Once you are familiar with the contents of these overview documents, you can proceed to the detailed documentation about the functionality provided by WebLogic Integration.

This document is organized as follows:

- Chapter 1, "Introduction," introduces the WebLogic Integration deployment architecture, including deployment resources, concepts, tasks, and the roles played by members of a deployment team.
- Chapter 2, "Understanding WebLogic Integration Clusters," describes how to deploy an integration solution on a cluster, which is a collection of servers that is managed as a single unit. It describes key clustering concepts and design tasks, and information about how a clustered deployment is configured.
- Chapter 3, "Configuring a Clustered Deployment," describes the steps you must take to set up and configure WebLogic Integration in a clustered environment.
- Chapter 4, "Understanding WebLogic Integration High Availability," describes how high availability is achieved for WebLogic Integration applications.
- Chapter 5, "Using WebLogic Integration Security," describes how to set up a secure WebLogic Integration deployment.
- Chapter 6, "Tuning Performance," describes key performance considerations in a WebLogic Integration deployment and explains how to monitor system performance. It provides instructions for tuning performance for WebLogic Integration resources, hardware, operating systems, network connectivity, and databases.

What You Need to Know

This document is intended primarily for:

- Deployment specialists who coordinate the deployment effort, designing the deployment topology for integration solutions, and configuring various WebLogic Integration features on one or more servers.
- System administrators who set up, deploy, and administer WebLogic Integration in a production environment.
- Database administrators who set up, deploy, and administer database management systems for WebLogic Integration in a production environment.

For an overview of the WebLogic Integration architecture, see *Introducing BEA WebLogic Integration*.

How to Print this Document

You can print a copy of this document from a Web browser, one file at a time, by using the File—>Print option on your Web browser.

A PDF version of this document is available on the WebLogic Integration documentation CD. You can open the PDF in Adobe Acrobat Reader and print the entire document (or a portion of it) in book format.

If you do not have the Adobe Acrobat Reader installed, you can download it for free from the Adobe Web site at http://www.adobe.com/.

Related Information

For information about installing WebLogic Integration and running the Configuration Wizard, see *Installing BEA WebLogic Platform* and *Using the Configuration Wizard*, which are available at the following URL:

`http://edocs.bea.com/platform/docs70/index.html`

WebLogic Integration documentation is available at the following URL:

`http://edocs.bea.com/wli/docs70/index.html`

WebLogic Server documentation is available at the following URL:

`http://edocs.bea.com/wls/docs70/index.html`

Contact Us!

Your feedback on the WebLogic Integration documentation is important to us. Send us e-mail at **docsupport@bea.com** if you have questions or comments. Your comments will be reviewed directly by the BEA professionals who create and update the WebLogic Integration documentation.

In your e-mail message, please indicate which version of the product and the documentation you are using.

When contacting Customer Support, be prepared to provide the following information:

- Your name, e-mail address, phone number, and fax number
- Your company name and company address
- Your machine type and authorization codes
- The name and version of the product you are using
- A description of the problem and the content of pertinent error messages

Documentation Conventions

The following documentation conventions are used throughout this document.

Convention	Item
Ctrl+Tab	Indicates that you must press two or more keys simultaneously.
italics	Indicates emphasis or book titles.

`monospace text`	Indicates code samples, commands and their options, data structures and their members, data types, directories, and filenames and their extensions. Monospace text also indicates text that you must enter from the keyboard.
	Examples:
	`#include <iostream.h> void main ( ) the pointer psz`
	`chmod u+w *`
	`\tux\data\ap`
	`.doc`
	`tux.doc`
	`BITMAP`
	`float`

`monospace boldface text`	Identifies significant words in code. *Example*: `void` **`commit`** `( )`
`monospace italic text`	Identifies variables in code. *Example*: `String` *`expr`*
UPPERCASE TEXT	Indicates device names, environment variables, and logical operators. *Example*s: LPT1 SIGNON OR
{ }	Indicates a set of choices in a syntax line. The braces themselves should never be typed.

Convention	Item

`[ ]`	Indicates optional items in a syntax line. The brackets themselves should never be typed.
	Example:
	`buildobjclient [-v] [-o name ] [-f file-list]...`
	`[-l file-list]...`

|	Separates mutually exclusive choices in a syntax line. The symbol itself should never be typed.

. . .	Indicates one of the following in a command line:
	■ That an argument can be repeated several times in a command line
	■ That the statement omits additional optional arguments
	■ That you can enter additional parameters, values, or other information
	The ellipsis itself should never be typed.
	Example:
	`buildobjclient [-v] [-o name ] [-f file-list]...`
	`[-l file-list]...`

. . .	Indicates the omission of items from a code example or from a syntax line. The vertical ellipsis itself should never be typed.

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
- Deployment Architecture
- Key Deployment Resources

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 1-1.
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 2, "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 3, "Configuring a Clustered Deployment."
4. Set up security for your WebLogic Integration deployment as described in Chapter 5, "Using WebLogic Integration Security."
5. Optimize overall system performance (once your WebLogic Integration deployment is running) as described in Chapter 6, "Tuning Performance."

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

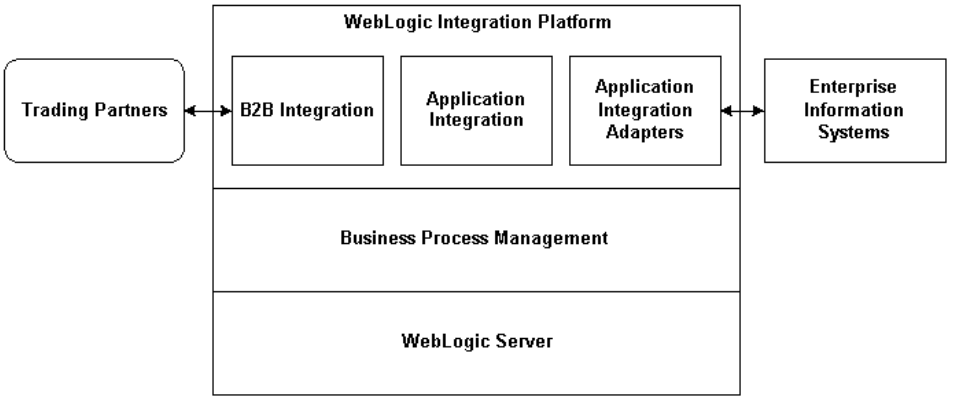
Database administrators provide in-depth technical and operational knowledge about database systems deployed in an organization. They have experience in the following areas:

- 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

Deployment Architecture

The following illustration provides an overview of the WebLogic Integration deployment architecture.

Figure 1-1 WebLogic Integration Deployment Architecture



The following section describes each of the resources illustrated in the preceding figure.

Key Deployment Resources

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

- WebLogic Server Resources
- Business Process Management Resources
- B2B Integration Resources
- Application Integration Resources
- Relational Database Management System Resources
- Hardware, Operating System, and Network Resources

WebLogic Server Resources

This section provides general information about BEA 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 2, "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 "Business Process Management Resources" on page 1-9.

For more information about WebLogic JMS, see the following topics:

- "Introduction to WebLogic JMS" in *Programming WebLogic JMS* at the following URL:

  `http://edocs.bea.com/wls/docs70/jms/intro.html`

- For more information about configuring and monitoring the JMS, see "Managing JMS" in the *BEA WebLogic Server Administration Guide* at the following URL:

  `http://edocs.bea.com/wls/docs70/adminguide/jms.html`

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. (For information about configuring pool and cache sizes, see "Configuring Other EJB Pool and Cache Sizes" on page 6-4.) The following table describes types of EJBs and their associated tunable parameter.

Table 1-1 Parameters for Tuning EJBs

Group Name	Description	Type of Resource Group
Event Listener Message-Driven Beans	`max-beans-in-free-pool`[1]	The maximum number of listeners that pull work from a queue.
Stateless Session Beans	`max-beans-in-free-pool`[1]	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		

1. The WebLogic Server documentation recommends setting the number of execute threads rather than setting `max-beans-in-free-pool`. However, in a WebLogic Integration environment, it is more efficient to control the workload by specifying the `max-beans-in-free-pool` setting of the event listener message-driven beans than by setting the number of execute threads.

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. For information about determining the size of a JDBC connection pool on each node in a WebLogic Integration cluster, see “Configuring JDBC Connection Pool Sizes” on page 6-5. 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* at the following URL:

  `http://edocs.bea.com/wls/docs70/jdbc/intro.html`

- "Managing JDBC Connectivity" in the *BEA WebLogic Server Administration Guide* at the following URL:

  `http://edocs.bea.com/wls/docs70/adminguide/jdbc.html`

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.

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. 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. Monitor your running system to empirically determine the best value for the execution thread pool.

For information about configuring the execution thread pool, see "Configuring the Execution Thread Pool" on page 6-7.

Following these recommendations for tuning your execution thread pool will help optimize the performance of WebLogic Integration. However, in a WebLogic Integration environment, the best way to throttle work is by controlling the number of message-driven beans—see "EJB Pooling and Caching" on page 1-6.

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, Proposed Final Draft 2, from Sun Microsystems, Inc.

For information about the WebLogic J2EE-CA, see "Managing the WebLogic J2EE Connector Architecture" in the *BEA WebLogic Server Administration Guide* at the following URL:

`http://edocs.bea.com/wls/docs70/adminguide/jconnector.html`

Business Process Management Resources

In WebLogic Integration, the Business Process Management (BPM) functionality handles the definition and execution of business processes. For an introduction to BPM functionality, see "Business Process Management" in *Introducing BEA WebLogic Integration*.

The following sections describe BPM features that are used for the deployment of WebLogic Integration solutions:

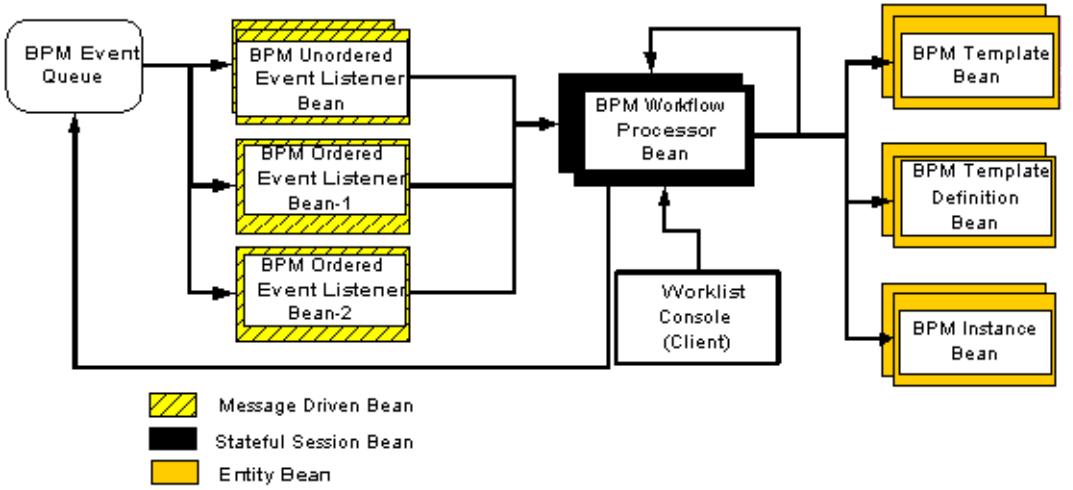
- Overview of BPM Resources
- Types of BPM Resources
- BPM Work Sequence

BPM resources can be configured to run on a cluster—a group of servers that is managed as a single unit. For more information about clustering and BPM, see Chapter 2, "Understanding WebLogic Integration Clusters."

Overview of BPM Resources

The following diagram shows BPM resources for a single node in a cluster.

Figure 1-2 BPM EJB Resources



The next section, "Types of BPM Resources," describes the resources represented in the preceding figure.

Types of BPM Resources

BPM uses WebLogic JMS (described in "Java Message Service" on page 1-6) for communicating worklist, time, and event notifications, as well as error and audit messages. BPM client applications send these messages, as *XML events*, to JMS event queues. BPM uses event listener message-driven beans to process XML events that arrive in event queues and deliver them to the running instance of the BPM engine.

You can create custom message queues using the WebLogic Server Administration Console, then run the MDB Generator utility to generate an event listener bean to listen on the queue, and subsequently update the BPM configuration to recognize the new event listener bean. For more information, see "Creating New Pools" on page 2-17.

The following sections describe the types of resources you can use when configuring BPM for a clustered environment and when tuning BPM performance:

- Workflow Processor Beans
- Event Listener Message-Driven Beans
- Template Beans
- Template Definition Beans
- Instance Beans
- Event Queue
- Worklist Console (Deprecated)

Workflow Processor Beans

Workflow processor beans are stateful session beans that execute workflows, which proceed from a start/event node to a stop/event node (quiescent state to quiescent state). Workflow processor beans accept work from event listener beans, Worklist clients, and from other workflow processor beans (when subworkflows are used).

Because workflow processor beans are instantiated at run time, based on the system load, the exact number of workflow processor beans at run time is dynamic. The size of the workflow processor bean pool determines the number of workflow processor beans that can be active concurrently. If the number of beans exceeds the pool size, then excess beans are passivated until a bean in the pool becomes available. In general, a pool size that is too large is preferable to one that is too small. For tuning information, see "Configuring Other EJB Pool and Cache Sizes" on page 6-4.

Workflow processor beans are deployed to the cluster. WebLogic Server optimizes a clustered system such that each node in a cluster uses a local copy of a workflow processor bean.

Event Listener Message-Driven Beans

Event listener message-driven beans pull work from the event queue and send work to the workflow processor beans. Event listener beans wait until the workflow processor bean either executes to completion or hits a quiescent state before getting new work from the queue.

Event listener beans have a configured pool size for unordered messages and they use a series of single bean pools (named beans with a free pool size of 1) for ordered messages, as described in "Generating Message-Driven Beans for Multiple Event Queues" in "Establishing JMS Connections" in *Programming BPM Client Applications*.

In combination, these pools determine the amount of parallel workflow execution that can occur when initiated from events.

Template Beans

Template beans are entity beans that contain the workflow template to be executed. In general, the size of the template entity bean pool should equal the maximum number of workflow templates (templates, not instances) to be executed concurrently. In general, a pool that is too large is preferable to one that is too small. Template entity beans are clusterable (they have cluster-aware stubs), so they can be used by workflow processor beans on other nodes in a cluster.

Template Definition Beans

Template definition beans are entity beans that contain the workflow template definition to be executed.

Business processes are saved as workflow templates in a database. These templates are essentially empty containers for storing different workflow versions.They can be associated with multiple organizations. Templates contain template definitions, which serve as different versions of the same workflow, and are distinguished by effective and expiry dates. For information about business processes and workflows, see *Using the WebLogic Integration Studio*.

In general, the size of the template definition entity bean pool should equal the maximum number of *workflow templates* (*not* workflow instances) to execute concurrently. In general, a pool that is too large is preferable to one that is too small. Template definition entity beans are clusterable (they have cluster-aware stubs), so they can be used by workflow processor beans on other nodes in a cluster.

Instance Beans

Instance beans are entity beans that contain the workflow instance being executed. In general, the size of the instance entity bean pool should equal the size of the workflow processor bean pool. There is no advantage to having an instance entity bean pool that

is larger than the workflow processor bean pool. In general, a pool that is too large is preferable to one that is too small. Instance entity beans are clusterable (they have cluster-aware stubs), so they can be used by workflow processor beans on other nodes in a cluster.

Event Queue

A single JAR file contains both ordered and unordered event listener message-driven beans for a particular queue. The WebLogic Integration installation provides the `wlpi-mdb-ejb.jar` file, which contains message-driven beans that consume messages from the default EventQueue. This JAR file must be targeted to the cluster. You can also create new event queues, as described in "Creating New Pools" on page 2-17. For information about BPM event queues in a cluster, see "Load Balancing BPM Functions in a Cluster" on page 2-15.

Note: To scale BPM functionality in a cluster, you must create new event queues.

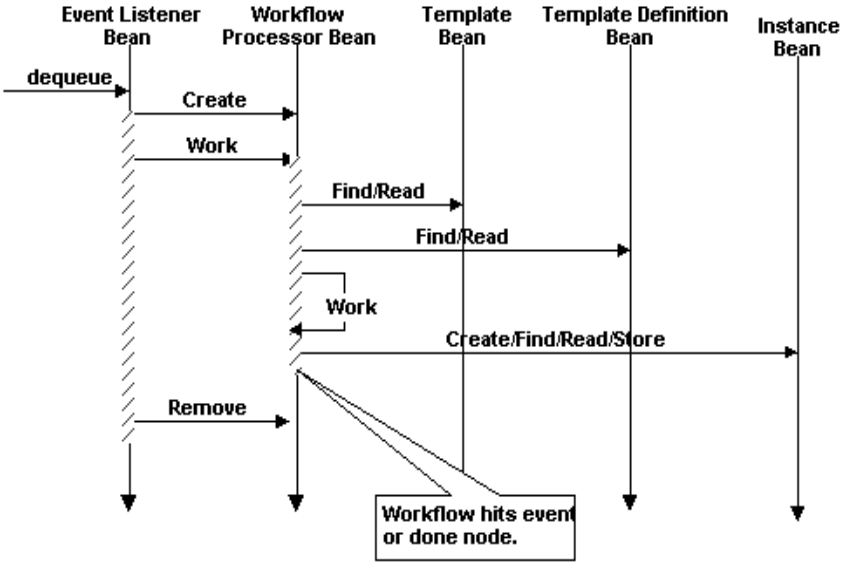
Worklist Console (Deprecated)

The Worklist client includes the swing-based WebLogic Integration Worklist console, as well as any user code that creates workflows from the BPM API. It is shown in Figure 1-2 for context only—it is not a configurable run-time resource.

BPM Work Sequence

The following diagram shows the interaction among BPM EJBs when processing events.

Figure 1-3 Interaction Between BPM EJBs When Processing Events



When a BPM *event listener bean* receives a work request from the event queue (whether the default queue or a user-defined queue), it creates a *workflow processor bean* to work on the request. The workflow processor bean executes the workflow until the workflow hits a stop or event node. Note that, when a workflow calls another workflow, a new workflow processor bean is created and the calling workflow does not exit the workflow processor bean.

The template bean and template definition bean are read at the beginning of workflow execution. The instance bean is read at the beginning of workflow execution, and written when workflow execution quiesces at a transaction boundary (such as an event or done node).

For event-driven workflows, the creation of additional workflow processor beans does not enable the deployment to do more work. The number of event listener beans limits the number of workflow instances that can be processed in parallel.

B2B Integration Resources

When you deploy WebLogic Integration to a clustered domain, all B2B integration resources, with the exception of resources for the administration server, must be deployed homogeneously in the cluster. That is, to achieve high availability, scalability, and performance improvements, B2B integration resources must be targeted to all clustered servers in a domain. For more information about B2B integration resources and clustering, see "Designing a Clustered Deployment" on page 2-3.

Many B2B integration resources are allocated dynamically, as needed; a deployment cannot be configured ahead of time. For information about resources that can be configured to accommodate B2B loads, see "Configuration Requirements" in *Administering B2B Integration*.

A shared file system is required for a cluster that uses B2B integration functionality. We recommend either a Storage Area Network (SAN) or a multiported disk system.

Note: WebLogic Integration applications that are based on the XOCP business protocol are not supported in a clustered environment.

Application Integration Resources

The following sections describe the types of application integration resources that WebLogic Integration supports:

- Synchronous Service Invocations
- Asynchronous Service Invocations
- Events
- Application Views and Connection Factories

For information about clustering and application integration, see Chapter 2, "Understanding WebLogic Integration Clusters."

Application integration functionality is integrated in the WebLogic Integration product, but it is also available packaged in a single, self-contained J2EE `ear` file. This enables you to deploy application integration on any valid WebLogic domain. For

example, Web services developers and WebLogic Portal developers can use application views to interact with EIS applications. For more information about deploying application integration outside of a WebLogic Integration environment, see "Modular Deployment of Application Integration" in *Using Application Integration*.

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 1-4 Synchronous Service Invocations

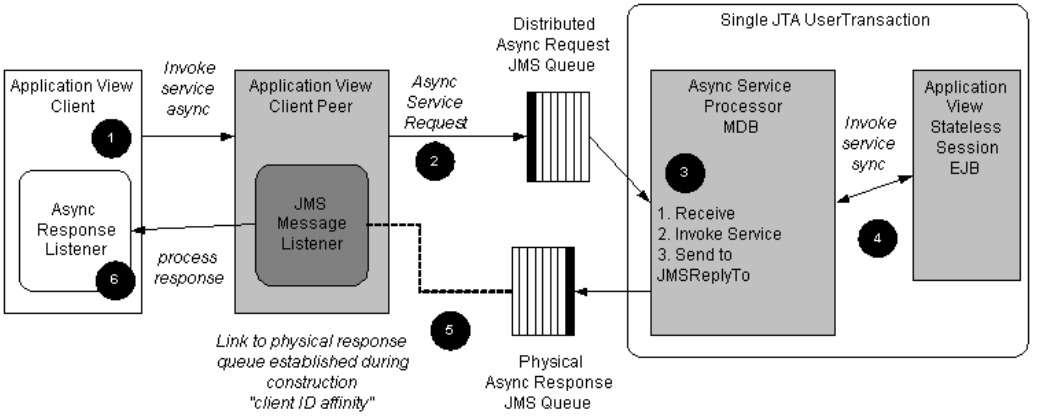
In a synchronous service invocation, a client (shown here as a *workflow processor*) calls the *application view EJB* (a stateless session bean). The application view calls the *service adapter* using a synchronous Common Client Interface (CCI) request. The service adapter is a J2EE-CA service adapter that actually processes the request.

Note: When a workflow acts as a client to an EIS, the workflow processor is stalled while it waits for the request to complete, tying up a workflow processor bean and perhaps an event listener bean as well. 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 1-5 Asynchronous Service Invocations



Note: The `WLAI_ASYNC_REQUEST_QUEUE` and `WLAI_ASYNC_RESPONSE_QUEUE` queues are deployed as distributed destinations in a WebLogic Integration cluster. The Asynchronous Service Request Processor is a message-driven bean (`wlai-asyncprocessor-ejb.jar`), which is also deployed to the cluster. For more information about how application integration resources are deployed for high availability, see "Highly Available JMS" on page 2-21.

The preceding diagram illustrates the following process flow for an application integration asynchronous service:

1. An Application View client instantiates an Application View instance.

   The client has the option of supplying a durable client ID at the time of construction. The durable client ID is used as the correlation ID for asynchronous response messages.

   The client invokes the `invokeServiceAsync` method and passes the request IDocument to an AsyncServiceResponse Listener to handle the response.

2. The Application View instance creates an `AsyncServiceRequest` object and sends it to the `WLAI_ASYNC_REQUEST_QUEUE`.

The AsyncServiceRequest object contains the name of the destination to which the response listener is pinned. The AsyncServiceProcessor message-driven bean uses this information to determine which physical destination to which it should send the response.

If a request object does not contain the name of a response destination, the AsyncServiceProcessor message-driven bean uses the destination specified for the JMS message (using a call to the JMSReplyTo() method).

Suppose, however, that only the client supplies an AsyncServiceResponseListener to the Application View:

```
invokeServiceAsync(String serviceName, IDocument request,
AsyncServiceResponseListener listener);
```

In this scenario, the Application View establishes a receiver to the JMS queue that is bound at the JNDI location provided by the Application View EJB method getAsyncResponseQueue*JNDIName*(). The Application View instance uses QueueReceiver.getQueue() to set the ReplyTo destination on the request message.

3. In a cluster, the WLAI_ASYNC_REQUEST_QUEUE queue is deployed as a distributed JMS queue. However, each message is sent to a single physical queue and is available only from that queue. If that physical queue becomes unavailable before a given message is dequeued, then the message (that is, the Asynchronous Service Request) remains unavailable until that physical queue comes back on-line via a manual JMS migration or server restart.

   It is not sufficient to send a message to a distributed queue and expect the message to be received by a QueueReceiver of that queue. Because the message is sent to only one physical queue, there must be a QueueReceiver listening on the physical queue. To satisfy this requirement, the AsyncServiceProcessor (wlai-asyncprocessor-ejb.jar) must be deployed on all nodes in a cluster.

   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, which in this case is `WLAI_ASYNC_RESPONSE_QUEUE_myserver1`.

   Note that the AsyncServiceProcessor must send the response to a specific physical destination (`WLAI_ASYNC_RESPONSE_QUEUE_myserver1`) and not to the distributed destination (`WLAI_ASYNC_RESPONSE_QUEUE`). The physical destination queue was established by the Application View instance running on the client when it called the Application View EJB `getAsyncResponseQueueJNDIName()` method. (See step 2.)

   Note: It is possible for a client application to fail before it receives all the response messages it expects. If, after recovery, you want to make sure that the client is associated with the same JMS response queue with which it was associated before the failure, you must use the same client ID that you used before the failure, after recovery. The following listing is an example of recovery code, which facilitates this association of the client with the JMS response queue by using the same unique client ID before the failure and after recovery:

```
String uniqueClientID = "uniqueClientID";

ApplicationView myAppView = new ApplicationView(jndiContext,
"MyAppView", uniqueClientID);

myAppView.recoverAsyncServiceResponses(new
MyAsyncResponseListener());
```

6. The instance of the Application View message listener that was created when the Application View instance was instantiated, 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 BPM or WebLogic Workshop. Events are forwarded from an Application View to a JMS queue (`WLAI_EVENT_QUEUE`). This queue is a distributed destination containing multiple physical destinations. A message-driven bean (the WLI-AI Event Processor) listens on the `WLAI_EVENT_QUEUE` distributed destination.

The WLI-AI Event Processor does the following:

- Delivers a copy of the event to the BPM event processor (if BPM is installed and running in the server instance) or to the Application View WebLogic Workshop Control event processor (if this control is being used).

  Exactly one copy of each event is delivered to BPM or WebLogic Workshop.

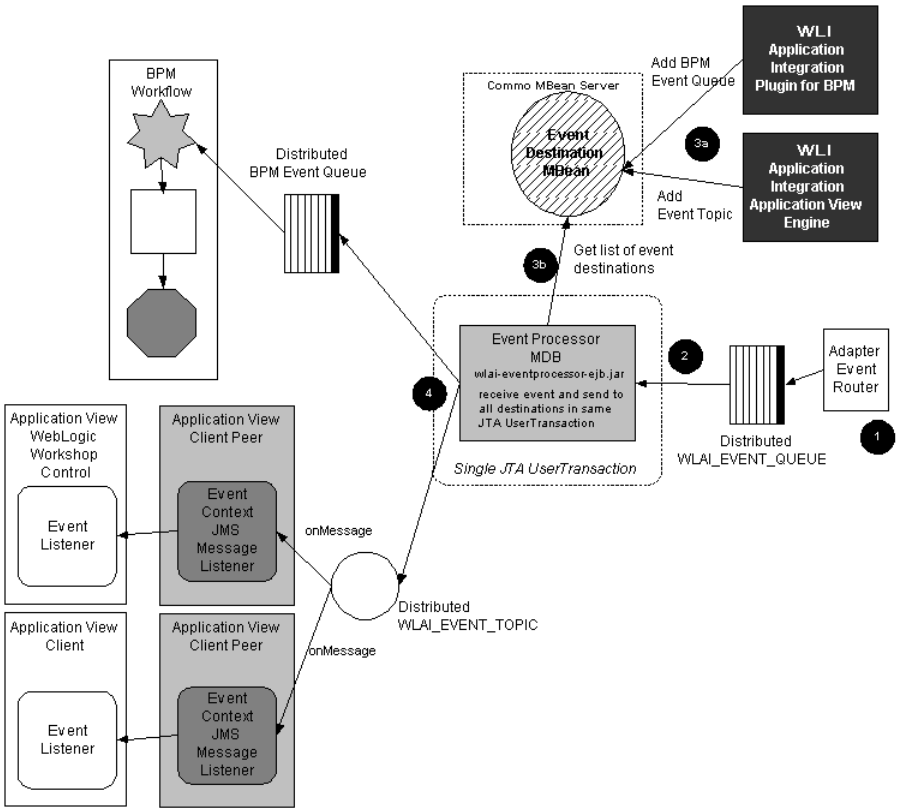
- Publishes a copy of the event to the `WLAI_EVENT_TOPIC`.

  The `WLAI_EVENT_TOPIC` is a distributed JMS topic and handles the delivery of events to remote Application View clients. The Application View class creates an EventContext on the `WLAI_EVENT_TOPIC`. The EventContext class filters messages based on the name of the Application View, which is stored in the `SourceKey` JMS header property. The `SourceType` is `ApplicationView`.

- Dequeues the event transactionally to allow the message to be rolled back to the queue if there is a processing failure.

The following figure illustrates event processing in WebLogic Integration.

Figure 1-6 Events



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

1. An event occurs in an enterprise information system (EIS) and is sent to a JMS queue, as follows:

   a. An event occurs in an enterprise information system (EIS).

   b. The event data is transferred to the event generator in the resource adapter. The event generator transforms the EIS-specific event data into an XML document and posts an IEvent object to the event router (Adapter Event Router).

c. The Event Router passes the IEvent object to an EventContext object for each application integration server that is interested in the specific event type.

d. The EventContext encapsulates the IEvent object into a JMS ObjectMessage and, using a JMS QueueSender, sends it to the JMS Queue bound at the following JNDI context: `com.bea.wlai.EVENT_QUEUE`.

2. The ObjectMessage is stored in the `WLAI_EVENT_QUEUE` and is processed by the WLI-AI Event Processor message-driven bean (`wlai-eventprocessor-ejb.jar`) in a first in, first out (FIFO) manner.

   In a cluster, `WLAI_EVENT_QUEUE` is deployed as a distributed JMS queue. However, each message is sent to a single physical queue and is only available from the physical queue to which it was sent. If that physical queue becomes unavailable before a given message is dequeued, then the message (that is, the event) is unavailable until that physical queue comes back on-line.

   It is not enough to send a message to a distributed queue and expect the message to be received by a QueueReceiver for that distributed queue. Because the message is sent to one physical queue, there must be a QueueReceiver listening on that physical queue. To satisfy this requirement, the WLI-AI Event Processor message-driven bean (`wlai-eventprocessor-ejb.jar`) must be deployed on all nodes in a cluster.

3. The WLI-AI Event Processor message-driven bean (`wlai-eventprocessor-ejb.jar`) determines the list of event destinations:

   a. Event destinations are added to the AIDestinationMBean. The MBean is replicated across the cluster so that the same list of event destinations is passed to the event processor message-driven bean on each managed server. When the WLI-AI BPM plug-in (`wlai-plugin-ejb.jar`) is deployed, it adds the BPM Event Queue as an event destination. The inclusion of this destination makes it possible for EIS events to be sent to the BPM process engine. Also, when Application View event listeners are registered, the event is sent to the `WLAI_EVENT_TOPIC`.

   b. The WLI-AI Event Processor message-driven bean reads the list of event destinations, to which it should send events, from the MBean.

4. An event ObjectMessage is delivered to all registered event destinations in a single JTA user transaction. If a message is not delivered to any event destination, it is rolled back on to the `WLAI_EVENT_QUEUE`. The `WLAI_EVENT_QUEUE` is

configured to forward poisoned messages to the WebLogic Integration error destination (`com.bea.wli.FailedEventQueue`). For information about the `FailedEventQueue`, see "Error Destination" on page 2-28.

Note: Because the destinations for events are typically JMS destinations, it is unlikely that the system will fail to forward an event.

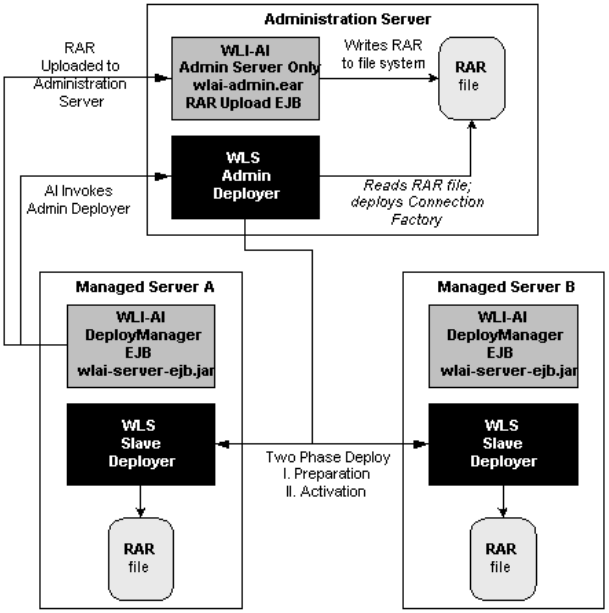
Application Views and Connection Factories

The run-time application integration features (synchronous service invocations, asynchronous service invocations, and events) described in preceding sections can be clustered for scalability and high availability. Design-time application integration features (Application Views and connection factories) can be clustered for scalability, but not for high availability. This means that you cannot deploy or undeploy (edit) Application Views if any server in a cluster is not running. In other words, you can deploy and undeploy (edit) only in a healthy cluster.

The resource adapter (RAR) is uploaded by deploying the `wlai-admin.ear` archive file to the administration server, not to the clustered managed servers. Two-phase deployment is used. The WebLogic Deployer utility on the administration server controls the deployment to managed servers.

The following figure illustrates connection factory deployment at design time.

Figure 1-7 Connection Factory Deployment



Application View deployment depends on successful connection factory deployment. For more information about deploying application integration adapters in clustered environments, see "Load Balancing Application Integration Functions in a Cluster" on page 2-19 and "Deploying Adapters" on page 2-30.

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 more information, see "Tuning Databases" on page 6-36.

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 *BEA WebLogic Integration Release Notes*. For more information about configuring these resources for maximum performance in a production environment, see "Recommended Hardware and Software" on page 4-2, and "Tuning Hardware, Operating System, and Network Resources" on page 6-32.

CHAPTER

2 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
- Understanding JMS Resources
- Deploying Adapters

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 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 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.

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* at the following URL:

  `http://edocs.bea.com/wls/docs70/cluster/index.html`

- "Understanding Cluster Configuration and Application Deployment" in *Using WebLogic Server Clusters* at the following URL:

  `http://edocs.bea.com/wls/docs70/cluster/config.html`

- "WebLogic Server Clusters and Scalability" in "Tuning WebLogic Server" in *BEA WebLogic Server Performance and Tuning* at the following URL:

  `http://edocs.bea.com/wls/docs70/perform/WLSTuning.html`

Designing a Clustered Deployment

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

- Introducing WebLogic Integration Domains
- Deploying WebLogic Integration Resources
- Load Balancing in a WebLogic Integration Cluster

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 Configuration Wizard that lets you generate domains from domain templates based on WebLogic Integration, business process management (BPM), or enterprise application integration (EAI) functionality. Based on user queries, the 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 *Configuration Wizard Template Reference*, which is available at the following URL:

`http://edocs.bea.com/platform/docs70/template/index.htm`

For information about creating WebLogic Integration domains using the Configuration Wizard, see Chapter 3, “Configuring a Clustered Deployment.”

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 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, see *Using WebLogic Server Clusters* in the WebLogic Server documentation set, at the following URL:

`http://edocs.bea.com/wls/docs70/cluster/index.html`

For details about basic, multi-tiered, and proxy architectures that are recommended, see "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 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 using the Servers node in the Administration Console.

This section describes WebLogic Integration resources and how they can be partitioned and distributed in a cluster. It contains the following topics:

- Clusterable Resources
- Two-Phase Deployment of WebLogic Integration
- Distribution Guidelines
- Deployment Order in WebLogic Integration Application
- Deploying The Default Web Application

- Note About Administration Servers

Clusterable Resources

Table 2-1 describes the WebLogic Integration deployment resources. It contains the following information:

- Resource Groups—Arbitrary designation of related deployment resources that are categorized for clustering purposes. All resources in a resource group must be targeted to the same machine.

  There are two types of resource groups:

  - *Clusterable*—Targeted to one or more servers, but all members of a group must be targeted to the same server or set of servers.
  - *Single Node*—Targeted to one and only one server in a cluster. Single Node resources must not be clustered.

- Resource Name—Name of an individual package or service (in a resource group) as it is shown in the WebLogic Server Console.

  Note that some resource names contain abbreviations that are a legacy from prior WebLogic Integration releases:

  - `wlc` corresponds to B2B integration
  - `wlpi` corresponds to BPM
  - `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 Administration Console.

The following table describes the WebLogic Integration deployment resources.

Table 2-1 WebLogic Integration Deployment Resources

Resource Group	Description (Single Node/Clusterable)	Resource Name	Administration Console Navigation
bpm-singleNode	BPM master components **(Single node)**	WLI-BPM Plugin Manager (`wlpi-master-ejb.jar`)	Deployments—EJB

Table 2-1 WebLogic Integration Deployment Resources (Continued)

Resource Group	Description (Single Node/Clusterable)	Resource Name	Administration Console Navigation
bpm-clusterable	BPM components **(Clusterable)**	WLI-BPM initialization (`bpm-init-ejb.jar`)	Deployments→EJB
		WLI-BPM Server (`wlpi-ejb.jar`)	Deployments→EJB
		WLI-BPM Event Processor MDBs (`wlpi-mdb-ejb.jar`)	Deployments→EJB
		User-defined Event Processor MDBs `wlpi-mdb-xxx.jar`[1]	Deployments→EJB
		wlpiFactory (`com.bea.wlpi.TopicConnectionFactory`)	Services→JMS→ Connection Factories
		wlpiQueueFactory (`com.bea.wlpi.QueueConnectionFactory`)	Services→JMS→ Connection Factories
		TXDataSource	Services→JDBC→ Tx Data Sources
B2B-singleNode	B2B integration administration **(Single node: Administration Server)**	B2B console (`b2bconsole.war`)	Deployments→ Web Applications
		WLI-B2B Startup (`b2b-startup.jar`) **Note:** Deployed to the administration server and the clustered managed servers.	Deployments→EJB

Table 2-1 WebLogic Integration Deployment Resources (Continued)

Resource Group	Description (Single Node/Clusterable)	Resource Name	Administration Console Navigation
B2B-clusterable	B2B integration components **(Clusterable)**	WLI-B2B Startup (`b2b-startup.jar`)	Deployments→EJB
		WLCShutdown	Deployments→ Startup & Shutdown
		WLCHub.DS	Services→JDBC→ Tx Data Sources
		TransportServlet (`b2b.war`)	Deployments→ Web Applications
		WLI-B2B RN MDB (`b2b-rosettanet.jar`)	Deployments→EJB
		WLI-B2B RN BPM Plug-in (`wlc-wlpi-plugin.jar`)	Deployments→EJB
		WLI-B2B ebXML BPM Plug-in (`ebxml-bpm-plugin.jar`)	Deployments→EJB
		RNQueueFactory (`com.bea.wli.b2b.rosettanet.QueueConnectionFactory`)	Services→JMS→ Connection Factories
		B2BTopicFactory (`com.bea.wli.b2b.server.TopicConnectionFactory`)	Services→JMS→ Connection Factories
AI-admin	Application integration administration (**Single node: Administration Server**)[2]	WLI-AI RAR Upload (`wlai-admin.ear`)	Deployments→ Applications→WLI-AI RAR Upload

Table 2-1 WebLogic Integration Deployment Resources (Continued)

Resource Group	Description (Single Node/Clusterable)	Resource Name	Administration Console Navigation
AI-clusterable	Application integration components **(Clusterable)**	WLI-AI Server (`wlai-server-ejb.jar`)	Deployments→EJB
		Application View Management Console (`wlai.war`)	Deployments→Web Applications→wlai
		WLI-AI Event Processor (`wlai-eventprocessor-ejb.jar`)	Deployments→EJB
		WLI-AI Async Processor (`wlai-asyncprocessor-ejb.jar`)	Deployments→EJB
		WLI-AI BPM Plug-in (`wlai-plugin-ejb.jar`)	Deployments→EJB
		WLI-AI BPM Plug-in Help (`wlai-plugin.war`)	Deployments→ Web Applications
		WLAI_JMSConnection Factory	Services→JMS→ Connection Factories
wlai-event-yyy[1]	Application integration event adapter **(Depends on the adapter[3])**	*yyyEventRouter*[1]	Deployments→ Applications→ *yyy*EventRouter[1,4]
wlai-service-yyy[1]	Application integration service adapter **(Depends on the adapter[3])**	*BEA . . . yyy . . . ADK_RAR*[1]	Deployments→ Applications→BEA . . . *yyy* . . . ADK_RAR[1,4]
		BEA . . . yyy . . . ADK_WEB[1]	Deployments→ Applications→BEA . . . yyy . . . ADK_WEB

Table 2-1 WebLogic Integration Deployment Resources (Continued)

Resource Group	Description (Single Node/Clusterable)	Resource Name	Administration Console Navigation
DI-clusterable	Data Integration components **(Clusterable)**	WLI-DI BPM Plug-in (`wlxtpi.jar`)	Deployments→EJB
		WLI-DI BPM Plug-in Help (`wlxtpi.war`)	Deployments→Web Applications
wli-clusterable	Resources that must be located on all servers in the domain **(Clusterable)**	WLI-Repository (`respository-ejb.jar`)	Deployments→EJB
		WLI Error Listener (`wli-errorlistener-mdb.jar`)	Deployments→EJB
		Mailsession (`wlpiMailSession`)	Services→Mail Java mail sessions used for the BPM Send E-mail action.
		JDBCConnectionPool (`wliPool`)	Services→JDBC→ Connection Pools Used for all database connections in WebLogic Integration.

1. Name represents a user-defined package or resource group.
2. You need to deploy `wlai-admin.ear` only when you deploy WebLogic Integration to a cluster; do not deploy it when you deploy in a single-node environment. For more information about the application integration administration component, see "Load Balancing Application Integration Functions in a Cluster" on page 2-19.
3. For example, the DBMS sample event adapter is deployed to a single node. For more information, see the documentation for the adapter you are using.
4. Event and service adapters reside in a single EAR file but they are deployed separately and are listed as separate resources in the WebLogic Server Administration Console. For more information, see the following section, "Two-Phase Deployment of WebLogic Integration.".

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 deployment of WebLogic Integration.

Listing 2-1 Deploying the WebLogic Integration Application

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

Distribution Guidelines

A WebLogic Integration cluster deployment conforms to the following guidelines:

- Most resources must be deployed to all servers in the cluster. For information about which WebLogic Integration resources are deployed to all managed servers in a cluster, a single managed server, and the administration server in a cluster, see "Deploying WebLogic Integration Resources" on page 2-4.
- Resources identified as members of the same resource group, as described in "Deploying WebLogic Integration Resources" on page 2-4, must be targeted to the same server, and, if those resources are identified as clusterable, they must be targeted to the same set of servers.
- The administration server does not require all WebLogic Integration resources, but you should deploy the following resources to it:
  - B2B Console (`b2bconsole.war`)
  - WLI-B2B Startup (`b2b-startup.jar`)
  - WLI-AI RAR Upload (`wlai-admin.ear`)
  - B2BTopic JMS Destination (`com.bea.wli.b2b.server.B2BTopic`)

- The number of JMS queues on a node should be determined using the guidelines described in "Load Balancing BPM Functions in a Cluster" on page 2-15 and "Load Balancing Application Integration Functions in a Cluster" on page 2-19. For information about how JMS resources are used in a WebLogic Integration deployment, see "Understanding JMS Resources" on page 2-23.

Targeting Resources to a Cluster

As shown in "Deploying WebLogic Integration Resources" on page 2-4, most WebLogic Integration resources are deployed to all the servers in a cluster. This deployment is specified in the configuration file (`config.xml`) for your domain.

You can use the WebLogic Server Administration Console to target components to nodes in your cluster. For more information, see Chapter 3, "Configuring a Clustered Deployment."

The following listing is an excerpt from the configuration file for a clustered domain, in which BPM components are specified. The listing shows how these components are targeted to a cluster named MyCluster.

Listing 2-2 Targeting WebLogic Integration Components to a Cluster

```
<Application Deployed="true" Name="WebLogic Integration"
    Path="C:/bea/weblogic700/integration/lib" TwoPhase="true">

<!--Repository-->

    <EJBComponent Name="WLI Repository" Targets="MyCluster"
     URI="repository-ejb.jar" />

<!--BPM-->

    <EJBComponent Name="WLI-BPM Server" Targets="MyCluster"
     URI="wlpi-ejb.jar" />

    <EJBComponent Name="WLI-BPM Event Processor"
     Targets="MyCluster" URI="wlpi-mdb-ejb.jar" />

    <EJBComponent Name="WLI-BPM Master Components"
     Targets="MyServer-1" URI="wlpi-master-ejb.jar" />

    <EJBComponent Name="WLI-BPM Initialization"
     Targets="MyCluster" URI="bpm-init-ejb.jar"/>
...
</Application>
```

In the preceding listing, note that all BPM components are targeted to the cluster, except the WLI-BPM master components (`wlpi-master-ejb.jar`). As specified in Table 2-1, the WLI-BPM master components must be deployed to one server in the cluster (in this case, MyServer-1).

Deployment Order in WebLogic Integration Application

The following file specifies all the components of WebLogic Integration:

WLI_HOME\lib\META-INF\application.xml

Because the components are deployed in the order in which they are listed in `application.xml`, you must not change the order in which they are listed in the file. The specified order is critical because it reflects dependencies among components. EJBs and BPM plug-ins are included in this application because they must be accessible to BPM functions.

If you deploy custom resources (such as custom plug-ins, EJBs, message-driven beans, and so on) to a WebLogic Integration application, you must edit the `application.xml` file to specify your new component.

Warning: You can specify a custom resource as the last entry in the `application.xml` file, unless your new resource is a plug-in to BPM, in which case, you must specify the new component as the penultimate (second to last) entry in the file. That is, it must be defined immediately before the `bpm-init-ejb.jar` module, but after all the other modules in the application.

The `bpm-init-ejb.jar` module must be the last module specified in `application.xml`:

```
<module>
    <ejb>bpm-init-ejb.jar</ejb>
</module>
```

Deploying The Default Web Application

By default, when you create a domain based on any of the WebLogic Integration domain templates, it contains configuration for a Web server deployed to the administration server. The Web server configuration, in turn, specifies the default Web application (`DefaultWebApp`).

The deployment descriptor (`web.xml`) for this default Web application resides in the following location:

DOMAIN_HOME\applications\DefaultWebApp_myserver\WEB-INF\

In the preceding line, *DOMAIN_HOME* represents the pathname of the domain you created.

A Web Application contains an application's resources, such as servlets, Java Server Pages (JSPs), JSP tag libraries, and any static resources such as HTML pages and image files.

Deploying Custom JSP and HTML Pages

If you deploy custom JSP or HTML pages as part of your WebLogic Integration application, your custom JSP and HTML pages should reside in the following directory:

DOMAIN_HOME\applications\DefaultWebApp_*node*

In the preceding path, *DOMAIN_HOME* represents the root directory of the custom domain you created using the Configuration Wizard (see "Step 2. Create a WebLogic Integration Domain" on page 3-7), and *node* represents the name of a WebLogic Server instance in your cluster.

You must configure a Web server for each node in your cluster. The following excerpt from a `config.xml` file shows:

- A `WebServer` element configured for a managed server named `managedserver1`
- An `Application` element configured for the default Web application

(Information of interest is in bold text for emphasis.)

Listing 2-3 WebServer Element for Managed Server in a config.xml File

```
<Server Name="managedserver1" ...
...
<WebServer Name="managedserver1" DefaultWebApp="DefaultWebApp_node"
    HttpsKeepAliveSecs="120" KeepAliveSecs="60"
    LogFileName="C:/bea/user_projects/mydomain/logs/access.log"
    LoggingEnabled="true"/>
```

```
...
</Server>

    :

<Application Deployed="true" Name="DefaultWebApp_node"
    Path="C:/bea/weblogic700/samples/integration/config/samples/RN2Security/
config/peer2/applications"
    StagedTargets="" TwoPhase="false">
    <WebAppComponent IndexDirectoryEnabled="true"
    Name="DefaultWebApp_node" Targets="managedserver1"
    URI="DefaultWebApp_node"/>

</Application>
```

In the preceding listing, note that the `DefaultWebApp` attribute in the `WebServer` element references the default Web application component. The configuration for the default Web application is also shown in the preceding listing. It, in turn, references the directory where your JSP and HTML pages reside (*`node`* represents the name of a server in your cluster).

For more information about deploying Web applications, see *Assembling and Configuring Web Applications*, which is available at the following URL:

`http://edocs.bea.com/wls/docs70/webapp/index.html`

Note About Administration Servers

If the administration server for a cluster is down, deployment or undeployment requests are interrupted, but managed servers should continue serving requests. You can boot or reboot managed servers using an existing configuration. However, you cannot change configuration for the cluster (for example, add new nodes to the cluster) until the administration server is recovered. For more information, see “Backup and Failover for an Administration Server” on page 4-18.

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 WebLogic Server Functions in a Cluster
- Load Balancing BPM Functions in a Cluster
- Load Balancing Application Integration Functions in a Cluster
- Load Balancing B2B Integration Functions in a Cluster

Load Balancing WebLogic Server Functions in a Cluster

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*, which is available at the following URL:

`http://edocs.bea.com/wls/docs70/cluster/index.html`

Load Balancing BPM Functions in a Cluster

BPM workflows require an event queue for processing event-based workflows. For more information, see "Business Process Management Resources" on page 1-9.

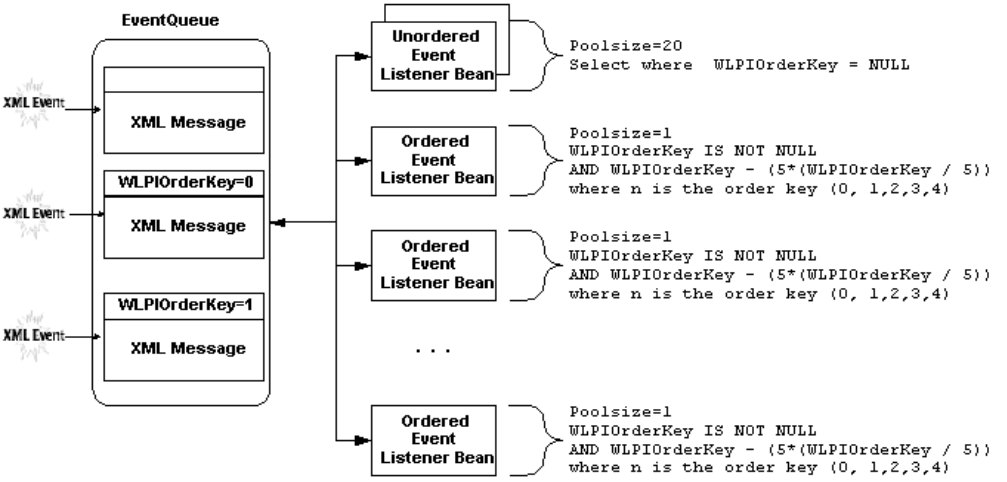
Event Queues and Associated Pools

The following types of pools are associated with each BPM event queue:

- Pool of *unordered event listener message-driven beans*
- Set of *ordered event listener message-driven beans* that select order keys from the JMS queue

The following figure illustrates an event queue and the pools associated with it.

Figure 2-1 Event Queue and Associated Pools



The *unordered event listener message-driven beans* process messages in a nondeterministic order. Although messages are read in first-in, first-out (FIFO) order, messages can be processed out of order after they are read, depending on thread scheduling and the load at the time they are processed.

The *ordered event listener message-driven beans* guarantee that, for a particular order key (`WLPIOrderKey`), messages are processed in an ordered sequence. To achieve this, a single event listener message-driven bean in a cluster must be configured to process messages for `WLPIOrderKey`.

An order key must be an integer value, and the value must be the same for each event that you want processed in the order in which it is received. Ordered messages must also be sent to the same JMS queue. The message producer is responsible for delivering the messages in the queue in the correct order.

`WLPIOrderKey` is a custom JMS property that BPM uses. You can set this property in the WebLogic Integration Studio or you can set it programmatically:

- You can set `WLPIOrderKey` in the post-XML event dialog box when you send messages between workflows. For more information, see "Posting an XML Message to a JMS Topic or Queue" in "Defining Actions" in *Using the WebLogic Integration Studio*.
- You can define the `WLPIOrderKey` JMS header field for a message programatically, as described in "Guaranteeing Sequential Processing of Messages" in "Establishing JMS Connections" in *Programming BPM Client Applications*.

A single JAR file (`wlpi-mdb-ejb.jar`) contains both ordered and unordered event listener message-driven beans for a particular queue. The message-driven beans provided in the `wlpi-mdb-ejb.jar` file consume messages from the default EventQueue. This JAR file must be targeted to the cluster.

BPM load balancing is achieved by deploying `wlpi-mdb-ejb.jar` to the cluster. This JAR file contains five ordered event listener message-driven beans and five unordered event listener message-driven beans. The message-driven beans consume messages from distributed destinations for validating and nonvalidating event queues. The distributed destinations contain one physical destination per JMS server, and one JMS server per 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). Making use of server affinity means that a message is kept within the same JVM and WebLogic Server instance while it is being processed. Therefore, ordered messages sent by a given producer to a distributed destination are guaranteed to be consumed by the same ordered message-driven bean. This process guarantees ordered delivery of messages.

Creating New Pools

If you have sufficient processing power on a single server, you can increase the pool size and range for event listener message-driven beans in the `wlpi-mdb-ejb.jar` file, as described in "Do You Have Enough Message-Driven Beans?" on page 6-20.

For information about creating a custom JMS queue and event listeners for that queue, see "Configuring a Custom Java Message Service Queue" in "Customizing WebLogic Integration" in *Starting, Stopping, and Customizing BEA WebLogic Integration*.

Requirements for Load Balancing BPM Functionality

When you are load balancing BPM functionality in a WebLogic Integration cluster, consider the following requirements:

- A JAR file containing the ordered and unordered event listener message-driven beans for a particular JMS queue must be targeted to the cluster. In other words, homogeneous deployment across the cluster is mandatory. WebLogic Integration provides the `wlpi-mdb-ejb.jar` file to pull work from the default event queue (`com.bea.wli.bpm.EventQueue`).

  Note: This description also applies to the validating queue when XML validation is being performed. The default validating event queue is `com.bea.wli.bpm.ValidatingEventQueue`.

- You can use the MDB Generator utility to create a new JAR file. The new JAR file must be associated with a new and unique JMS queue.
- Applications must be aware of the new JMS queue in order to trigger work on the new event listener message-driven beans.

Timed Events

Like the message-driven beans for the event and validating event queues, timed event listeners are also deployed to the cluster in the `wlpi-mdb-ejb.jar` file. These message-driven beans pull work from `com.bea.wli.bpm.TimerQueue`.

Timed events are implemented using JMS delivery times. They are executed by two types of pools:

- A pool of timed event listener message-driven beans similar to the pool for non-ordered processing (timed start workflows).
- A set of message-driven beans for consuming the timed events with which a workflow instance (task due date, timed event action) is associated. (Single timed event listener message-driven beans in a cluster are configured to process messages for specific order keys.)

Load Balancing Application Integration Functions in a Cluster

It is possible to configure a homogenous cluster (that is, one in which all resources have the same managed server targets), subject to any constraints in the adapters themselves.

In contrast to BPM functionality, it is possible to load balance application integration functionality in a cluster using the default JMS queues and servers.

In a clustered deployment, you must deploy a single EJB (`wlai-admin-ejb.jar`) to the administration server only. This EJB is deployed from the `wlai-admin.ear` archive file. (See Table 2-1 in "Deploying WebLogic Integration Resources" on page 2-4.)

Note: The `wlai-admin-ejb.jar` is required only in a clustered deployment. Therefore do not deploy `wlai-admin.ear` when you deploy WebLogic Integration to a single server.

The following code excerpt is from a `config.xml` file for a clustered domain. It shows the deployment specification for the `wlai-admin.ear` file in the cluster.

Listing 2-4 Deploying the wlai-admin EJB in the config.xml File

```
<Application Name="WLI-AI Admin Server Only"
Path="WLI_HOME/lib/wlai-admin.ear" TwoPhase="true">

<EJBComponent Name="WLI-AI RAR Upload" Targets="admin_server_name"
URI="wlai-admin-ejb.jar"/>

</Application>
```

Load Balancing B2B Integration Functions in a Cluster

B2B integration functionality does not require partitioning of work within a cluster; to support such functionality, you should configure a completely homogenous cluster. In other words, all B2B resources (JMS consumers, destinations, and producers) are available on all nodes in the cluster.

It is possible to load balance B2B integration functionality in a cluster using the default JMS queues and servers.

B2B integration resources are deployed homogeneously to all nodes in a cluster. Therefore, when shutdown of the B2B engine is requested, the B2B engine on all nodes in a cluster are shut down. It is not possible to shut down the B2B engine on a single node in a cluster; you must first remove the node from the cluster.

By using distributed destinations, WebLogic JMS balances the messaging load across multiple physical destinations, which can result in better use of resources and improved response times. The WebLogic JMS load-balancing algorithm determines the physical destinations (in a distributed destination set) to which messages are sent, as well as the physical destinations to which consumers are assigned. Message-driven beans are bound to the physical destination in the server on which they are deployed (server affinity).When a message is sent to a particular physical destination (or queue) on a particular server, the message is processed by that server.

B2B integration functionality takes advantage of the server affinity heuristic and in-memory caching in a clustered environment. During B2B message processing, the B2B decoder enqueues the message envelope for a B2B message into the BPM JMS event queue. A BPM message-driven bean dequeues the message and a B2B-specific plug-in is invoked to further process the message. The B2B-specific plug-in uses the message ID, the trading partner, and the delivery channel (URI) to retrieve the message payload from the MessageStore in-memory cache. Therefore, B2B integration functions can make use of in-memory caching, resulting in improved performance.

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 "JMS Servers and JMS Destinations" on page 2-25.

Highly Available JMS

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.

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. However, destinations deployed as singletons are less desirable because the migration required for them is manual.

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). Therefore, ordered messages sent by a given producer to a distributed destination are guaranteed to be consumed by the same ordered message-driven bean. This process guarantees ordered delivery of messages, and makes possible the B2B caching described in "Load Balancing B2B Integration Functions in a Cluster" on page 2-20.

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.

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
- High Availability for Event Forwarding

High Availability for Asynchronous Service Requests

`WLAI_ASYNC_REQUEST_QUEUE` and `WLAI_ASYNC_RESPONSE_QUEUE` queues are deployed as distributed destinations in a WebLogic Integration cluster, and the Asynchronous Service Request Processor is the associated message-driven EJB, which is deployed to all servers in a cluster. Asynchronous requests and responses are processed even after the JMS server that accepted them crashes.

If a physical queue fails before an asynchronous service request is received by a message-driven bean, the request remains unavailable until the physical queue comes back on line. The same scenario is true for asynchronous service responses.

For information about processing of synchronous and asynchronous invocations for application integration functions, see “Application Integration Resources” on page 1-15.

High Availability for Event Forwarding

Application integration adapters generate events that are consumed by BPM functionality or by WebLogic Workshop. Events are forwarded from an adapter to a JMS queue (`WLAI_EVENT_QUEUE`).

To obtain metadata about events, an event router communicates with a WebLogic Integration instance, using HTTP. If you want to achieve load balancing and high availability for event router callback communication, but you are not using a DNS name for your cluster address, you must set the `wlai.clusterFrontEndHostAndPort` property. For information about this property, see “Setting the wlai.clusterFrontEndHostAndPort Property (Optional)” on page 3-5.

The WLAI_EVENT_QUEUE is a distributed destination containing multiple physical destinations. A message-driven bean (the AI Event Processor) listens on the WLAI_EVENT_QUEUE distributed destination. Because multiple servers participate in the processing of messages for this queue, a single server failure can be accommodated. For information about how adapter events are processed by WebLogic Integration, see "Events" on page 1-19.

Understanding JMS Resources

This section describes how to configure JMS resources for your WebLogic Integration application in a clustered environment. Specifically, it describes how to configure the following resources:

- JMS Connection Factories
- JMS JDBC Stores
- JMS Servers and JMS Destinations
- Creating a Store and Associating It with a Connection Pool
- Creating a JMS Server and Associating It with the Store

JMS resources are configured in the WebLogic Server Administration Console. To start the console, see "Starting the WebLogic Server Administration Console" in "WebLogic Integration Administration and Design Tools" in *Starting, Stopping, and Customizing BEA WebLogic Integration*.

JMS Connection Factories

The following JMS connection factories are configured in a WebLogic Integration domain that contains an administration server and clustered managed servers:

- One for a BPM topic; deployed to the cluster
- One for a BPM queue; deployed to the cluster
- One for an application integration queue; deployed to the cluster
- One for a B2B integration topic; deployed to the administration server only

- One for the B2B RosettaNet queue; deployed to the cluster

The following listing shows a sample specification for deployment of JMS connection factories in a WebLogic Integration cluster. Note the Targets and JNDI names for the connection factories, which are shown in bold.

Listing 2-5 JMSConnectionFactory Elements in the config.xml File

```
...
<!--Application Integration Connection Factories>

<JMSConnectionFactory AllowCloseInOnMessage="false"
    DefaultDeliveryMode="Persistent" DefaultPriority="4"
    DefaultTimeToLive="0"
    JNDIName="com.bea.wlai.JMSConnectionFactory"
    MessagesMaximum="10" Name="WLIJMSConnectionFactory"
    OverrunPolicy="KeepOld" Targets="MyCluster"
    UserTransactionsEnabled="true"/>

<!--B2B Integration Connection Factories>

<JMSConnectionFactory AllowCloseInOnMessage="true"
    JNDIName="com.bea.wli.b2b.server.TopicConnectionFactory"
    Name="B2BTopicFactory" Targets="MyServer-1"
    UserTransactionsEnabled="true"/>

<JMSConnectionFactory AllowCloseInOnMessage="true"
    JNDIName="com.bea.wli.b2b.rosettanet.QueueConnectionFactory"
    Name="RNQueueFactory" Targets="MyCluster"
    UserTransactionsEnabled="true"/>

<!--BPM Connection Factories>

<JMSConnectionFactory AllowCloseInOnMessage="true"
    JNDIName="com.bea.wlpi.TopicConnectionFactory"
    Name="wlpiFactory" Targets="MyCluster"
    UserTransactionsEnabled="true"/>

<JMSConnectionFactory AllowCloseInOnMessage="true"
    JNDIName="com.bea.wlpi.QueueConnectionFactory"
    Name="wlpiQueueFactory" Targets="MyCluster"
    UserTransactionsEnabled="true"/>

...
```

JMS JDBC Stores

A JMS JDBC store must be defined for each JMS server in your deployment.

The following listing is an excerpt from a `config.xml` file, which defines JMS JDBC stores for a cluster (`MyCluster`) containing two managed servers (`MyServer-1` and `MyServer-2`), administered by the `myserver` administration server. Note that the target for the connection pool lists both the cluster and the administration server.

Listing 2-6 JMSJDBCStore Elements in the config.xml File

```
<JMSJDBCStore ConnectionPool="wliPool"
    Name="JMSWLCStore-MyServer-1" PrefixName="MyServer_1"/>

<JMSJDBCStore ConnectionPool="wliPool"
    Name="JMSWLCStore-MyServer-2" PrefixName="MyServer_2"/>

<JMSJDBCStore ConnectionPool="wliPool" Name="JMSWLCStore-myserver"
    PrefixName="myserver"/>

...

<JDBCConnectionPool CapacityIncrement="1"
DriverName="oracle.jdbc.driver.OracleDriver" InitialCapacity="1"
LoginDelaySeconds="1" MaxCapacity="15" Name="wliPool"
Properties="user=scott;password=tiger;dll=ocijdbc8;protocol=thin
RefreshMinutes="0" ShrinkPeriodMinutes="15"
ShrinkingEnabled="true" Targets="myserver,MyCluster"
URL="jdbc:oracle:thin:@machine:port:name"/>
```

JMS Servers and JMS Destinations

One JMS server must be configured for each managed server in your cluster, and one for the administration server. (Only one destination (the B2B Topic) is deployed to the JMS server configured for the administration server, as shown in Table 2-2.) We recommend the following naming convention for your JMS servers: `WLI_JMSServer_`*`node`*, such that *`node`* represents the name of the server on which the JMS server is deployed.

The following table describes the destinations (JMS queues and topics) used by WebLogic Integration and specifies whether they are deployed as single or distributed destinations.

Table 2-2 JMS Destinations

Destination	Distributed or Single
com.bea.wli.bpm.TimerQueue	Distributed
com.bea.wli.bpm.EventQueue	Distributed
com.bea.wli.bpm.ValidatingEventQueue	Distributed
com.bea.wli.bpm.ErrorTopic	Distributed
om.bea.wli.bpm.AuditTopic	Distributed
com.bea.wli.bpm.NotifyTopic	Distributed
com.bea.wlpi.EventTopic	Single managed server
com.bea.wli.b2b.server.B2BTopic	Administration server only
com.bea.b2b.OutboundQueue	Distributed
com.bea.b2b.rosettanet.EncoderQueue	Distributed
com.bea.wlai.ASYNC_REQUEST_QUEUE	Distributed
com.bea.wlai.ASYNC_RESPONSE_QUEUE	Distributed
com.bea.wlai.EVENT_QUEUE	Distributed
com.bea.wlai.EVENT_TOPIC	Distributed
com.bea.wli.FailedEventQueue[1]	Distributed

1. The com.bea.wli.FailedEventQueue destination is used by all components of WebLogic Integration. It should be used as the error destination for any JMS destination that consumes messages in a JTA UserTransaction. For more information about the error queue, see “Error Destination” on page 2-28.

The following listing is an excerpt from a `config.xml` file. It shows selected JMS elements for a clustered configuration containing two managed servers (`MyServer-1` and `MyServer-2`), administered by an administration server (`myserver`).

Listing 2-7 JMSServer Elements in the config.xml File

```
<!--Distributed Destinations-->

<JMSDistributedQueue JNDIName="com.bea.wli.bpm.EventQueue"
    Name="WLI_BPM_Event" Targets="MyCluster">

    <JMSDistributedQueueMember JMSQueue="WLI_BPM_Event_MyServer-1"
    Name="WLI_BPM_Event_MyServer-1"/>
    <JMSDistributedQueueMember JMSQueue="WLI_BPM_Event_MyServer-2"
    Name="WLI_BPM_Event_MyServer-2"/>

</JMSDistributedQueue>

<!--Administration Server-->

<JMSServer Name="WLI_JMSServer_myserver"
    Store="JMSWLCStore-myserver" Targets="myserver"
    TemporaryTemplate="TemporaryTemplate">
    <JMSTemplate Name="TemporaryTemplate"/>
    <JMSTopic JNDIName="com.bea.wli.b2b.server.B2BTopic"
    Name="B2BTopic"/>
</JMSServer>

<!--Managed Server-->

<JMSServer Name="WLI_JMSServer_MyServer-1"
Store="WLI_JMSJDBCStore_MyServer-1" Targets="MyServer-1
(migratable)"

    <JMSQueue JNDIName="com.bea.wli.bpm.Event.MyServer-1"
    Name="WLI_BPM_Event_MyServer-1" StoreEnabled="true"
    Template="WLI_JMSTemplate-1"/>

...
    <JMSTopic JNDIName="com.bea.wli.bpm.EventTopic"
    Name="wlpiEvent" StoreEnabled="false"/>

...
</JMSServer>
```

Note the following information in the preceding listing:

- One JMS distributed destination (JMSDistributedQueue) is shown. It specifies the WLI_BPM_Event queue:
  - The queue is deployed to all servers in the cluster, as specified by the following: `Targets="MyCluster"`
  - The JMSDistributedQueue element contains two JMSDistributedQueueMember elements, one for each physical destination. That is, one destination is associated with each managed server.
- Two JMSServer elements are shown: one for the administration server and one for a managed server.
- The JMSServer element for the administration server contains one JMSTopic element, specifying that only one destination (the B2B Topic destination) is deployed on the administration server.
- The JMSServer for each managed server should contain JMSTopic and JMSQueue elements for all the JMS distributed destinations described in Table 2-2. The excerpt from the `config.xml` file shown in this listing illustrates the following examples:
  - The JMSQueue element specifying the WLI_BPM_Event queue for MyServer-1 (`WLI_BPM_Event_MyServer-1`).
  - The JMSTopic element specifying `com.bea.wli.bpm.EventTopic`, which must be deployed to only one server in the cluster (in this case, `MyServer-1`).

Error Destination

The `com.bea.wli.FailedEventQueue` is the error destination for any JMS destination that consumes messages in a JTA User Transaction. For example, it is the error destination for the EventQueue, ValidatingEventQueue, and TimerQueue. Messages that do not find a target BPM workflow instance and those that fail a number of retries, at one minute intervals, are sent to the FailedEventQueue. (The default number of retry attempts is 10, but you can configure a different number.) When a JMS message arrives on the FailedEventQueue, a message-driven bean (`com.bea.wli.common.errorlistener.ErrorListenerBean`), which listens on the queue, writes log entries to the WebLogic Server log.

You can specify the number of retry attempts to be allowed by configuring attributes for redelivery in the `WLI_JMSTemplate-node`, which is the JMS template used by the error queue. The error queue is a distributed destination, and the redelivery attributes are configured for a node-specific physical destination, which is named `WLI-FailedEvent-node`. (For these names, *node* represents the name of a WebLogic Server instance in your cluster):

1. In the WebLogic Server Administration Console navigation tree, choose Services—JMS—Templates—WLI_JMSTemplate-*node*.
2. Select the Configuration tab, followed by the Redelivery tab.
3. Specify the redelivery attributes and the appropriate WLI-FailedEvent-*node* for which to configure those attributes.
4. Click Apply.

For more information about configuring redelivery attributes for a JMS template, see "JMS Template—Configuration—Redelivery" in "JMS" in the *Adminstration Console Online Help*, which is available at the following URL:

`http://e-docs.bea.com/wls/docs70/ConsoleHelp/domain_jmstemplate_config_redelivery.html`

In addition, you have the option of creating your own custom message listener, adding it to the classpath, and referencing it in the FailedEventQueue message-driven bean deployment descriptor. By doing so, you can configure your system to persist error messages.

Creating a Store and Associating It with a Connection Pool

To create a store and associate it with a connection pool, complete the following steps:

1. In the Administration Console navigation tree, go to the Services—JMS—Stores node and select Configure a new JMSJDBCStore. The Configuration tab should be selected by default.
2. In the Name field, enter the name by which you want to identify this store.

   Every JMS server has its own JMSJDBCStore. Every managed server has its own JMS server. For instructions on creating such a server, see "Creating a JMS Server and Associating It with the Store" on page 2-30.
3. In the Connection Pool field, select the connection pool that you want to use.

4. In the Prefix Name field, enter the prefix to be appended (for example, WLI-AI).
5. Click Create.

Creating a JMS Server and Associating It with the Store

To create a JMS server and associate it with a JMSJDBCStore, complete the following steps:

1. In the Administration Console navigation tree, go to the Services—JMS—Servers node and select Configure a new JMSServer.
2. In the Name field, enter the name by which you want to identify this JMS server.
3. In the Store field, select the JMSJDBCStore with which you want to associate this JMS server.
4. In the Temporary Template field, select one of the available templates:
   - Temporary Template
   - WLIJMSTemplate

   Note: You can access the properties for each of these JMS templates through the Services—JMS—Templates node in the Administration Console.
5. Click Create.

Deploying Adapters

Run-time application integration features (synchronous service invocations, asynchronous service invocations, and events), described in "Application Integration Resources" on page 1-15, can be clustered for scalability and high availability. Application integration features that are available at design time (Application Views and Connection Factories) can be clustered for scalability but not for high availability, meaning that you cannot deploy or undeploy (edit) Application Views if any server in a cluster is not running. In other words, you can deploy and undeploy (edit) only in a healthy cluster.

An application integration adapter is typically composed of three components:

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

The resource adapter (RAR) file and the design time Web application (WAR) file should be deployed to the cluster. However the event generator Web application (WAR) file should, in most cases, be deployed to a single node in the cluster. (For specific information, see the documentation for the adapter you are using.)

For example, when a DBMS adapter is used, the DbmsEventRouter Web application must be targeted to a single node in a cluster. The following listing is an excerpt from a `config.xml` file. It shows an Application element that specifies the configuration for deployment of a DBMS adapter.

Listing 2-8 Configuration for Deploying an Application Integration Adapter

```
<Application Deployed="true" Name="BEA_WLS_DBMS_ADK"
Path="/bea/weblogic700/integration/adapters/dbms/lib/
BEA_WLS_DBMS_ADK.ear" StagingMode="stage" TwoPhase="true">

    <ConnectorComponent Name="BEA_WLS_DBMS_ADK"
Targets="MyCluster" URI="BEA_WLS_DBMS_ADK.rar"/>

    <WebAppComponent Name="BEA_WLS_DBMS_ADK_Web"
Targets="MyCluster" URI="BEA_WLS_DBMS_ADK_Web.war"/>

    <WebAppComponent Name="DbmsEventRouter" Targets="MyServer-1"
URI="BEA_WLS_DBMS_ADK_EventRouter.war"/>
</Application>
```

Note the following information in the preceding listing. Information of interest is highlighted in bold text for emphasis:

- The value of StagingMode must be set to `stage`, to ensure that the administration server copies the adapter EAR file to all managed servers in the StagedTargets list before activating the server.
- Adapters must be deployed using the TwoPhase=”true” attribute. This setting ensures that all adapter components are deployed before your system attempts to process messages.

- The resource adapter design-time Web application is deployed to the cluster, but the event generator Web application, deployed from a WAR file, is deployed to only one managed server (`MyServer-1`).

Configuring Adapters for Deployment

WebLogic Integration domains created by the Configuration Wizard define the configuration for resource adapters. In the configuration created by the Configuration Wizard, the three components of an adapter are targeted for deployment to the cluster. However, as described in the preceding section (specifically, Listing 2-8), the event generator Web application (WAR) file should, in most cases, be deployed to a single node in the cluster. You must modify your domain configuration to meet this requirement. For information about how to modify the domain configuration, see "Step 5. Configure Event Router WAR File for Adapters" on page 3-15.

You can also deploy resource adapters after you start the servers in your cluster. For information about how to set up and start your clustered deployment, see Chapter 3, "Configuring a Clustered Deployment." 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 B, "Deploying Resource Adapters."

For more information about deploying adapters in the WebLogic Integration environment, see "Deploying Adapters" in *Developing Adapters*.

CHAPTER

3 Configuring a Clustered Deployment

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

For information about deploying WebLogic Integration on a single server, see "Creating and Customizing a New Domain" in "Customizing WebLogic Integration" in *Starting, Stopping, and Customizing BEA WebLogic Integration*.

Plan the architecture of your clustered domain, as described in "Designing a Clustered Deployment" on page 2-3; then 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 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. Create a WebLogic Integration Domain
- Step 3. Configure the Database for Your Domain
- Step 4. Configure BPM Resources for One Managed Server
- Step 5. Configure Event Router WAR File for Adapters
- Step 6. Configure an RDBMS Realm
- Step 7. Configure a Router

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 Configuration Wizard. (See "Step 2. Create a WebLogic Integration Domain" on page 3-7.)

  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 a listen addresses for servers when you create a WebLogic Integration domain using the Configuration Wizard. (See "Step 2. Create a WebLogic Integration Domain" on page 3-7.)

  - 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.

    In this case, 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"/>
```

Note: For development and testing, you can use a comma-separated list. However, for production, it is recommended that you specify the cluster address only as a DNS name or a single IP address. If you do not use a DNS name for the ClusterAddress, you should set the `wlai.clusterFrontEndHostAndPort` property to support load balancing and high availability for event router callback communication (application integration functionality). For information about setting this property, see "Setting the wlai.clusterFrontEndHostAndPort Property (Optional)" on page 3-5.

- Configure an Oracle, Microsoft SQL, Sybase, or DB2 database for your clustered domain.
- Include a shared file system. A shared file system is strongly recommended for any cluster in which B2B integration functionality is used. 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.
- Configure a hardware or software router for your system. Load balancing of servlets and JSPs can be accomplished using either the built-in load balancing capabilities of a WebLogic proxy plug-in, or separate load balancing hardware.

  For information about configuring a software router for your WebLogic Integration domain, see "Step 7. Configure a Router" on page 3-18.

  For information about hardware and software routers, see *Using WebLogic Server Clusters*, which is available at the following URL:

  `http://edocs.bea.com/wls/docs70/cluster/index.html`

For more information about setting up clustered WebLogic Server instances, see "Setting Up WebLogic Clusters" in *Using WebLogic Server Clusters*, which is available at the following URL:

`http://e-docs.bea.com/wls/docs70/cluster/index.html`

Note: Additional requirements apply when you design your domain to include one or more firewalls. For details, see "Communications in a Cluster" in *Using WebLogic Server Clusters*, which is available at the following URL:

`http://edocs.bea.com/wls/docs70/cluster/index.html`

Setting the wlai.clusterFrontEndHostAndPort Property (Optional)

Application integration adapters generate events that are consumed by the BPM engine. For information about events and event processing in WebLogic Integration, see "Events" on page 1-19.

When you do not use a DNS name for your cluster address, you should set the `wlai.clusterFrontEndHostAndPort` property to achieve load balancing and high availability for event router callback communication.

Why Set the wlai.clusterFrontEndHostAndPort Property?

The following table describes the cluster configuration in a sample cluster, for which the cluster address is:

```
<Cluster ClusterAddress="127.0.0.1:7001,127.0.0.1:7002"
Name="MyCluster"/>
```

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
MyRouter	Router	127.0.0.1:7003

An event router uses obtains metadata about events by communicating with an instance of WebLogic Integration through HTTP. Exactly how such communication occurs is determined by whether or not the `wlai.clusterFrontEndHostAndPort` property is set:

- If the `wlai.clusterFrontEndHostAndPort` property is *not* set . . .

  As a result, upon establishment of communication with the event router, WebLogic Integration passes the first address listed in `ClusterAddress` as the callback address. In our example, the callback address is `127.0.0.1:7001`. In this scenario, if MyServer-1 fails, then event routers cannot contact the WebLogic Integration application, even though MyServer-2 is still running.

- If the `wlai.clusterFrontEndHostAndPort` property *is* set . . .

  The `wlai.clusterFrontEndHostAndPort` property is set to the address of the cluster front-end, in this case, the MyRouter server, which hosts the HttpClusterServlet.

  As a result, upon establishment of communication with the event router, WebLogic Integration passes the following address to the event router: `127.0.0.1:7003`. Even if a managed server in the cluster fails, event routers can still contact the WebLogic Integration application in this scenario.

How to Set the wlai.clusterFrontEndHostAndPort Property

You must create a `wlai.clusterFrontEndHostAndPort` property in the WLAIStartup EJB environment properties for each managed server. For example, to set `wlai.clusterFrontEndHostAndPort=127.0.0.1:7003`, complete the following procedure:

1. In the Administration Console navigation tree, choose *Domain_Name*—Deployments—EJB—WLI-AI Server.
2. Click Edit EJB Descriptor to display a new window, in which you can edit the EJB descriptor.
3. In the left navigation pane of the new window, choose EJB Jar—Enterprise Beans—Sessions—WLAIStartup—Env Entries to display a configuration window.
4. Click Configure a New Environment Entry.

5. Enter the following information:
   - Env Entry Name: `wlai.clusterFrontEndHostAndPort`
   - Env Entry Value: *`127.0.0.1:7003`*

     `127.0.0.1:7003` represents the listen address and port for the cluster front-end (in this case, the router) that hosts the HttpClusterServlet.

Step 2. Create a WebLogic Integration Domain

To complete this step, 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.

You begin the definition of a clustered WebLogic Integration deployment by creating a domain using the BEA Configuration Wizard.

Note: The procedure described in this section for setting up your domain is based on the assumption that you are running the Configuration Wizard in GUI mode from the Windows Start menu.

For information about using the Configuration Wizard in different modes, see *Using the Configuration Wizard*, which is available at the following URL:

`http://edocs.bea.com/platform/docs70/confgwiz/index.html`

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

1. From the Start Menu, choose Programs→BEA WebLogic Platform 7.0→Domain Configuration Wizard.

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

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

In this window . . .	Perform the following action . . .
Choose a Domain Type and Name	Select a template on which to base your domain and assign a name to your domain. Select one of the following domain templates, depending on the requirements for your domain: ■ WLI Domain—Use this template when you want your domain to support all WebLogic Integration functionality: BPM, B2B integration, application integration, and data integration. ■ WLI BPM—Use this template when you want your domain to support BPM and data integration functionality. ■ WLI EAI—Use this template when you want your domain to support application integration, BPM, and data integration functionality.
Choose Server Type	Select the following option when you are prompted for the server type: `Admin Server with Clustered Managed Server(s)`
Choose Domain Location	Specify a directory in which to install your domain. Accept the default directory or browse to select a different one. Any valid directory on your machine can be used as the domain directory.
Configure Clustered Servers	Specify the server name, listen address, and listen port for each managed server in your cluster.[1]
Configure Cluster	Specify the cluster name, cluster multicast address, cluster multicast port, and cluster address.[1]
Configure Standalone/Administrative Server	Specify the server name, server listen address, server listen port, and server SSL listen port for the administration server, from which all administrative functions for your clustered domain will be performed.[1] **Note:** When you configure the administration server, we recommend that you accept the default Server Name (`myserver`), as prompted by the Configuration Wizard. If you specify a server name other than the default, you must change the name of the following directory in your domain, by replacing `myserver` with the new name you specified: *`DOMAIN_HOME`*`/applications/DefaultWebApp_`*`myserver`* In the preceding path, *`DOMAIN_HOME`* represents the root directory of the custom domain you created using the Configuration Wizard.
Create Administrative User	Specify a user name and password.

Create Start Menu Entry for Server	Specify whether you want to install the administration server in the Windows Start menu.
Configuration Summary	Do one of the following: ■ Review the configuration summary information and click Create to create the defined domain. ■ Review the configuration summary information and click Previous to return to windows you have already viewed, so you have an opportunity to change information that you supplied earlier, before creating the domain.

1. For information about setting up addresses and port numbers, see "Step 1. Comply with Configuration Prerequisites" on page 3-2.

When you complete the domain configuration using the Configuration Wizard, your new domain is created in the location you specified earlier. A configuration file (`config.xml`) is created in the domain. It contains a definition for the administration server and each managed server in the cluster, and it assigns the managed servers to the cluster.

Note: In the steps that follow in this procedure, you will edit the `config.xml` file to configure your clustered domain. Therefore, we recommend that before proceeding to the next step, you save a backup copy of the `config.xml` file created in this step.

Step 3. Configure the Database for Your Domain

The Database Wizard is a WebLogic Integration configuration utility that facilitates the task of setting up a database for the domain you created in the preceding step.

To run the Database Wzard:

1. Run the `wliconfig` script in the domain you created in step 2.

   If, for example, you created a domain named `mydomain` in the default location, run one of the following command sequences (depending on your operating system):

- Windows:

```
cd %BEA_HOME%\user_projects\mydomain
wliconfig
```

- UNIX:

```
cd $BEA_HOME/user_projects/mydomain
wliconfig
```

2. The Database Wizard provides the following options:

   - Switch Database
     Select this option to designate a different database to be used for the domain. The environment variables used by commands that are invoked to initialize the database (commands such as `CreateDB` and `RunSamples`) are updated, and the `config.xml` file is modified to reflect the new settings. This option does not initialize the database; rather, it configures the environment in preparation for database initialization.
   - Create Database
     Select this option to initialize the database currently specified, or to switch to a new database and initialize it.

3. The Database Wizard prompts you to provide the values required to connect to the database you are configuring.

For information about running the Database Wizard, see "Using the Database Wizard" in "Customizing WebLogic Integration" in *Starting, Stopping, and Customizing BEA WebLogic Integration*.

Step 4. Configure BPM Resources for One Managed Server

As described in "JMS Servers and JMS Destinations" on page 2-25, and in "Deploying WebLogic Integration Resources" on page 2-4, the following BPM resources must be deployed to a single node in your cluster:

- The WLI-BPM Plugin Manager, for which the URI is `wlpi-master-ejb.jar`.
- The EventTopic JMS topic, for which the JNDI name is `com.bea.wlpi.EventTopic`.

You must modify your domain configuration to meet this requirement. You can do this in one of two ways:

- Edit the Configuration File
- Use the WebLogic Server Administration Console

Edit the Configuration File

The `config.xml` file in your domain contains comments to help you quickly identify and edit the elements that must be changed. Complete the following steps to make the required changes to the domain's configuration file:

1. In the root directory of the domain you created, open the `config.xml` file in a text editor.
2. Search for comments labeled with the word MODIFY.

   The following table shows the elements in the configuration file labeled with a MODIFY comment, and describes the modifications you are required to make.

Table 3-1 Configuring BPM Resources for One Managed Server

Locate the element . . .	To modify the element . . .

<!-- MODIFY: In a cluster, the BPM Plugin Manager Targets attribute must specify only ONE cluster server -->
`<EJBComponent Name="WLI-BPM Plugin Manager"`
`Targets="`**`mycluster`**`"`
`URI="wlpi-master-ejb.jar"/>`
	Change the Targets attribute to deploy the WLI-BPM Plugin Manager EJB component to a managed server, represented, in this case, by *`managedserver-1:`*
	
	`<EJBComponent Name="WLI-BPM Plugin Manager"`
	`Targets="`*`managedserver-1`*`"`
	`URI="wlpi-master-ejb.jar"/>`

<!-- MODIFY: This JMS Topic must be deployed to only one node in the cluster. Uncomment this section for one node.
`<JMSTopic Name="wlpiEvent"`
`JNDIName="com.bea.wlpi.EventTopic"`
`/>`
-->
	Uncomment this JMSTopic element for one, and only one, managed server in the cluster configuration:
	
	`<JMSTopic Name="wlpiEvent"`
	`JNDIName="com.bea.wlpi.EventTopic"/>`

3. Proceed to "Step 5. Configure Event Router WAR File for Adapters" on page 3-15.

Use the WebLogic Server Administration Console

If you start the administration server in your domain before making the changes described in the preceding table, comments that were written in the `config.xml` file are lost. However, if the comments are lost, you can still configure your system appropriately, by following the procedures described in the following sections:

- Configure BPM Master EJB for One Managed Server
- Configure BPM Event Topic for One Managed Server

Configure BPM Master EJB for One Managed Server

The simplest method to configure the BPM master EJB for one managed server is to modify the domain configuration file, as described in Table 3-1.

Note: The following procedure is provided for reference only. Use it, for example, if the comments in your `config.xml` file are lost before you complete the configuration.

To configure the BPM Master EJB for a single managed server:

1. In the Administration Console navigation tree, select the WLI-BPM Plugin Manager EJB in the domain you created:

   Domain_Name—Deployments—EJB—WLI-BPM Plugin Manager

2. Select the Targets tab and follow the instructions in the Administration Console to change the deployment settings in such a way that the WLI-BPM Plugin Manager EJB is deployed to only one managed server in your cluster.

   For more information about using the Administration Console to configure the deployment of EJBs, see the *WebLogic Server Adminstration Console Online Help*, which is available at the following URL:

   `http://e-docs.bea.com/wls/docs70/ConsoleHelp/index.html`

After you create your domain, following the procedures in "Step 2. Create a WebLogic Integration Domain" on page 3-7, the WLI-BPM Plugin Manager is targeted to the cluster. In other words, the Targets attribute contains the cluster name.

After you complete the required modification, the WLI-BPM Plugin Manager EJB should be targeted to a single managed server in your cluster. The following excerpt shows the edited `config.xml` file, which illustrates the configuration change in the Targets attribute. In this example, the EJB is targeted to a managed server named `manageserver-1`.

Listing 3-1 Targeting WLI-BPM Plugin Manager to a Managed Server

```
<Application Name="WebLogic Integration" Deployed="false"
Path="C:/bea/weblogic700/integration/lib" TwoPhase="true">

...
<EJBComponent Name="WLI-BPM Plugin Manager"
Targets="manageserver-1"
URI="wlpi-master-ejb.jar"/>

...
</Application>
```

Configure BPM Event Topic for One Managed Server

The simplest method to complete your domain configuration is to modify the domain configuration file, as described in "Step 4. Configure BPM Resources for One Managed Server" on page 3-11.

Note: The following procedure is provided for reference only. Use it, for example, if the comments in your `config.xml` file are lost before you complete the configuration.

To configure a BPM Event Topic for a single managed server:

1. In the Administration Console navigation tree, select Destinations for a JMS server. For example, choose:

   Services→JMS→Servers→WLIJMSServer_*manageserver-1*→Destinations

   WLIJMSServer_*manageserver-1* represents the name of the JMS server that is specific to one managed server in your domain.

   Note: You can assign any name you choose to the JMS server. However, we recommend that you follow the naming convention used in this example: `WLIJMSServer_`*`node`*. When this convention is used, *`node`* represents the name of the server on which the JMS server is deployed.

2. Click Configure a New JMS Topic to display the Configuration tab.
3. Enter the following information in the appropriate fields:
   - Name: `wlpiEvent`
   - JNDI Name: `com.bea.wlpi.EventTopic`
4. Accept the defaults for the remainder of the fields, and click Apply.

For more information about creating new JMS queues and topics, see the *WebLogic Server Adminstration Console Online Help*, which is available at the following URL:

`http://e-docs.bea.com/wls/docs70/ConsoleHelp/index.html`

The following listing is an excerpt from a `config.xml` file to which the BPM Event Topic is added to the JMSServer element for a managed JMS server. (In this case, the managed server is named `WLIJMSServer_manageserver1`.) The section of code being highlighted in this listing is shown in bold.

Listing 3-2 Configuration for com.bea.wlpi.EventTopic

```
<JMSServer Name="WLIJMSServer_manageserver1"
      Targets="manageserver1 (migratable)"
      TemporaryTemplate="TemporaryTemplate"
      Store="JMSWLIStore_manageserver1">

      <JMSTopic Name="wlpiEvent"
JNDIName="com.bea.wlpi.EventTopic"/>

...
</JMSServer>
```

Step 5. Configure Event Router WAR File for Adapters

Note: This step is required only for domains that are based on the WebLogic Integration or the EAI domain templates. If your clustered domain is based on the BPM domain template, proceed to "Step 6. Configure an RDBMS Realm" on page 3-17.

As described in "Deploying WebLogic Integration Resources" on page 2-4, the event router WAR files for sample adapters must be deployed on a single node in a cluster. To meet this requirement, you must edit your domain configuration for the adapters that are configured as part of your WebLogic Integration domain or your EAI domain: BEA_WLS_DBMS_ADK and BEA_POWERENTERPRISE_3_0 adapters.

To change your configuration, you can use the WebLogic Server Administration Console, or you can edit the `config.xml` file in your domain.

Using the Administration Console

1. In the Administration Console navigation tree, select the following event router WAR files in the domain you created:
   - *Domain_Name*→Deployments→WebApplications→DbmsEventRouter
   - *Domain_Name*→Deployments→WebApplications→ BEA_POWERENTERPRISE_3_0_EventRouter
2. Change the deployment settings for each event router by selecting the appropriate Targets tab and following the instructions in the Administration Console. Specify that each event router will be deployed to only one managed server in your cluster.

   For more information about using the Administration Console to configure the deployment of Web applications, see the *WebLogic Server Adminstration Console Online Help*, which is available at the following URL:

   http://e-docs.bea.com/wls/docs70/ConsoleHelp/index.html

Using the config.xml File

The following listing is an excerpt from a sample `config.xml` file for a domain containing an administration server with clustered managed servers. It shows the event router WAR files for two adapters, configured such that they are deployed on a single managed server in the cluster. The `WebAppComponent` elements are shown in bold.

Listing 3-3 Configuring BEA_WLS_DBMS_ADK and BEA_POWERENTERPRISE_3_0 Adapters

```
<Application Deployed="true" Name="BEA_WLS_DBMS_ADK"
Path="<WLI_HOME>/adapters/dbms/lib/BEA_WLS_DBMS_ADK.ear"
TwoPhase="true">

      <ConnectorComponent Name="BEA_WLS_DBMS_ADK"
        Targets="MyCluster" URI="BEA_WLS_DBMS_ADK.rar"/>

      <WebAppComponent Name="DbmsEventRouter" Targets="MyServer-1"
        URI="BEA_WLS_DBMS_ADK_EventRouter.war"/>
```

```
      <WebAppComponent Name="BEA_WLS_DBMS_ADK_Web"
        Targets="MyCluster" URI="BEA_WLS_DBMS_ADK_Web.war"/>

</Application>

 :

<Application Deployed="true" Name="BEA_POWERENTERPRISE_3_0"
Path="<WLI_HOME>/adapters/powerenterprise/lib/
BEA_POWERENTERPRISE_3_0_EAR.ear" TwoPhase="true">

      <ConnectorComponent Description="J2EE CA adapter for
        PowerEnterprise!" Name="BEA_POWERENTERPRISE_3_0"
        Targets="MyCluster" URI="BEA_POWERENTERPRISE_3_0.rar"/>

        <WebAppComponent
Name="BEA_POWERENTERPRISE_3_0_EventRouter"
        Targets="MyServer-1"
        URI="BEA_POWERENTERPRISE_3_0_EventRouter.war"/>

        <WebAppComponent Name="BEA_POWERENTERPRISE_3_0_Web"
        Targets="MyCluster"
URI="BEA_POWERENTERPRISE_3_0_Web.war"/>

</Application>
```

Step 6. Configure an RDBMS Realm

If your domain uses an RDBMS Realm from a previous release of WebLogic Integration, you must include an RDBMSRealm element in the `config.xml` file for your domain. The element is configured, but it is disabled in the `config.xml` file generated when you create your domain using the procedures in "Step 2. Create a WebLogic Integration Domain" on page 3-7. To enable the RDBMSRealm element:

1. Open the `config.xml` file in the root directory of your WebLogic Integration domain.
2. Search for the RDBMSRealm element and uncomment the RDBMS Realm element.

3. The RDBMSRealm element provided for you in the `config.xml` file is configured for the Pointbase database, as shown in the following listing. If you are using another database, reconfigure the DatabaseDriver, DatabasePassword, DatabaseURL, and DatabaseUserNameattributes in the RDBMSRealm element.

Listing 3-4 RDBMSRealm Element

```
<RDBMSRealm Name="wlpiRDBMSRealm"
            DatabaseDriver="com.pointbase.jdbc.jdbcUniversalDriver"
            DatabasePassword="none"
            DatabaseURL="jdbc:pointbase://localhost:9094/WLIDB"
            DatabaseUserName="none"

            :
            :
```

For more information about migrating security realm data, see "Step 8. Migrate Your Security Realm Data" in "Migrating WebLogic Integration 2.1 to WebLogic Integration 7.0" in the *BEA WebLogic Integration Migration Guide*.

Step 7. Configure a Router

If you want to configure a software router based on the built-in WebLogic `HttpClusterServlet`, you can do so by uncommenting a predefined section in the `config.xml` file in your domain.

The `config.xml` file in your domain contains comments to help you quickly identify and edit the elements that must be changed. Complete the following steps to make the required changes to the domain's configuration file:

1. Open the `config.xml` file in the root directory of your WebLogic Integration domain.
2. Search for comments labeled with the words ROUTER-OPTION, and follow the instructions included in those comments to configure a router for your domain.

You must provide appropriate values for the router server name, listen address, listen port, and so on.

3. Configuration for a Web server is included in the router configuration, as shown in the following code from the `config.xml` file:

```
<WebServer Name="ROUTER_NAME"
DefaultWebApp="DefaultWebApp_ROUTER_NAME"
. . .  />
```

Note that the Web server element references the default Web application through the `DefaultWebApp` attribute. (*`ROUTER_NAME`* represents the name you assign to the router.)

a. Make sure that a directory matching the specified DefaultWebApp value (in this case, `DefaultWebApp_`*`ROUTER_NAME`*) resides in the following location:

`DOMAIN_HOME``/applications/`

In the preceding path, *`DOMAIN_HOME`* represents the root directory for your domain.

b. Create a `web.xml` deployment descriptor in the DefaultWebApp directory specified in the preceding step. The `web.xml` deployment descriptor should include the registration of the HttpClusterServlet.

For information about creating a web.xml deployment descriptor, see "Configure Proxy Plug-Ins" in "Setting up WebLogic Clusters" in *Using WebLogic Server Clusters*, which is available at the following URL:

`http://edocs.bea.com/wls/docs70/cluster/index.html`

Note: When you configure a hardware or software router for your cluster, messages coming from outside the cluster should be sent to the URL of the router.

For information about hardware and software routers, see *Using WebLogic Server Clusters*, which is available at the following URL:

`http://edocs.bea.com/wls/docs70/cluster/index.html`

Step 8. Edit the startWeblogic Command File

You must edit the `startWeblogic.cmd` or `startWeblogic.sh` file in your domain to set the `-Dweblogic.management.discover` parameter to `true`:

1. Open the `startWeblogic` command file in the root directory of your WebLogic Integration domain.
2. Locate the `-Dweblogic.management.discover` argument to the start WebLogic command.
3. Change the value specified from `false` to `true`.

   The following code listing shows an example of a start server command and includes the modification to the `-Dweblogic.management.discover` argument. This code listing represents a single command. It is shown here as multiple lines for the sake of readability. In your command file, however, it is entered as one physical line.

Listing 3-5 Start Server Command for a WebLogic Integration Clustered Domain

```
REM Start weblogic

%JAVA_HOME%\bin\java %JAVA_VM% %JAVA_OPTIONS% -Xmx256m
-classpath %SVRCP%
-Dweblogic.servlet.ClasspathServlet.disableStrictCheck=true
-Dwli.bpm.server.evaluator.supportsNull=false
-Dweblogic.management.username= -Dweblogic.management.password=
-Dweblogic.Name=adminserver
-Dweblogic.RootDirectory=%WLI_DOMAIN_HOME%
-Djava.security.policy=%WL_HOME%\lib\weblogic.policy
-Dweblogic.management.discover=true
-Dweblogic.ProductionModeEnabled=%STARTMODE%  weblogic.Server
```

By making

For a scenario in which the administration server is restarted while managed servers are running in your domain, the administration server can discover running managed servers when `-Dweblogic.management.discover` is set to true.

Step 9. Set Up Managed Servers for Your Domain

This step provides instructions for extending the domain you created by adding managed servers. To add a managed server to a domain, you must create the managed server and configure WebLogic Integration components on the server.

A WebLogic Integration domain can be set up in one of the following ways:

- The administration server and the clustered managed servers can be set up on the same machine.
- The administration server and the clustered managed servers can be set up on different machines.
- Any combination of the preceding two configurations. For example, a cluster might comprise one machine that hosts an administration server and a number of managed servers, and one or more other machines that host additional managed servers.

This section provides instructions for setting up managed servers in your cluster:

- Add a Managed Server to an Existing Installation
- Add a Managed Server in a New Location

Both procedures explain how to add a managed server to a domain created with one of the templates provided with the Configuration Wizard.

Add a Managed Server to an Existing Installation

Complete the following steps to add a managed server to a WebLogic Integration domain:

- Step 1. Create a New Managed Server
- Step 2. Update the Domain Configuration for the New Managed Server (Optional)

Note: The procedures in this section are based on the assumption that your domain is named `mydomain`, and that it resides in the default location: *`BEA_HOME`*`\user_projects`.

Step 1. Create a New Managed Server

1. Start the administration server and the WebLogic Server Administration Console:
   a. To start the administration server, see "Starting WebLogic Integration" in "Getting Started" in *Starting, Stopping, and Customizing BEA WebLogic Integration*.
   b. To start the console, see "Starting the WebLogic Server Administration Console" in "WebLogic Integration Administration and Design Tools" in *Starting, Stopping, and Customizing BEA WebLogic Integration*.
2. In the Administration Console navigation tree, select Servers.
3. Click Configure a New Server.
4. Enter values in the Name, Listen Address (server instance IP address) fields and, if applicable, in the external DNS name field.

   The value you specify for the external DNS name can be a host name, or a virtual host name for a multihomed machine. (A multihomed machine is one for which multiple IP addresses are assigned.)
5. Select the machine name from the Machine drop-down list.
6. Click Create.
7. Select the Cluster tab.
8. Select the appropriate cluster from the Cluster drop-down list.

Note: For more information about creating and configuring servers, clusters, machines, and domains using the WebLogic Server Administration Console, see the *WebLogic Server Adminstration Console Online Help*, which is available at the following URL:

`http://edocs.bea.com/wls/docs70/ConsoleHelp/index.html`

Step 2. Update the Domain Configuration for the New Managed Server (Optional)

You must add a managed server to the configuration for a domain before you can start that server. You may add a managed server to the domain configuration when you are creating the domain, or you may add it after a domain has been created. Therefore, this step is optional. Use the following guidelines to determine whether you need to complete it:

- You must update your domain configuration if the managed server you created in the previous step ("Step 1. Create a New Managed Server" on page 3-22) was not added to your configuration when you initially created your domain. (In other words, updating is required if you did not define the managed server in the Configuration Wizard when you performed "Step 2. Create a WebLogic Integration Domain" on page 3-7.)
- You do not need to update your domain configuration if the managed server you created in "Step 1. Create a New Managed Server" on page 3-22 is already configured for your domain. (In other words, updating is not required if you defined the managed server in the Configuration Wizard when you created your domain, as described in "Step 2. Create a WebLogic Integration Domain" on page 3-7.)

To update your domain configuration for a new managed server, complete the following procedure:

1. Configure a JMS JDBC Store, and associate it with a connection pool:
   a. In the Administration Console navigation tree, choose *Domain_Name*—Services—JMS—Stores
   b. Click Configure a New JMSJDBCStore to display the Configuration tab.
   c. Enter the following information in the appropriate fields:

      Name: `JMSWLIStore_new`*`manageserver`*

      Connection Pool: `wliPool`

      Prefix Name: *`newmanageserver`*

   d. Click Create to create a new JMSJDBCStore for the new managed server.

2. Configure a JMS Server and associate it with the JMS JDBC Store:

   a. In the Administration Console navigation tree, choose *Domain_Name*—Services—JMS—Servers

   b. Click Configure a New JMS Server to display the Configuration tab.

   c. Enter the following information in the appropriate fields:

      Name: `WLIJMSServer_new`*`manageserver`*

      Store: `JMSWLIStore_new`*`manageserver`*

      Temporary Template: `TemporaryTemplate`

      `WLIJMSServer_`*`newmanageserver`* represents the name of the new JMS server. `JMSWLIStore_new`*`manageserver`* is the name you gave the JMSJDBC Store when you created it.

      You can give the JMS server any name you choose. However, we recommend using the naming convention in this example: `WLIJMSServer_`*`node`*. In this format, *`node`* represents the name of the server (WebLogic Server instance) on which the JMS server is deployed.

   d. Accept the defaults for the remainder of the fields, and click Create to create a new JMS server for your new managed server.

3. Configure destinations for the newly defined JMS server:

   a. Access your new JMS server configuration in the Administration Console navigation tree:

      Domain_Name—Services—JMS—Servers—*Server_Name*—Destinations

   b. Click Configure Destinations.

   c. Click Configure a new JMS Topic or Configure a new JMS Queue.

      Configure the `WLI_FailedEvent-`*`node`* destination first. By doing so, a `WLI_JMSTemplate-`*`node`* can be configured for redelivery. (For information about how to configure redelivery attributes for physical destinations, see “Error Destination” on page 2-28.) The `WLI_JMSTemplate-`*`node`* is used by several of the other queue destinations.

      For information about using the Administration Console to complete this task, see “JMS Destination Tasks” in “JMS” in the *WebLogic Server*

Adminstration Console Online Help, which is available at the following URL:

`http://edocs.bea.com/wls/docs70/ConsoleHelp/index.html`

Note: You should refer to an existing node in the cluster to see the destinations that need to be configured, and which destinations use the `WLI_JMSTemplate-node`. The destinations required vary depending on the domain template you used when you created your domain.

For a list of all JMS queues and topics configured for a JMS server in a domain based on the WLI Domain template, see "JMS Servers and JMS Destinations" on page 2-25.

4. Configure WebLogic Integration distributed destinations.

   Multiple physical JMS destinations are configured for every distributed destination. One physical destination is configured for each managed server in your cluster.

   To configure JMS destinations for a newly created managed server, complete the following steps:

   a. Access the distributed destinations for your WebLogic Integration domain by selecting the following nodes in the Administration Console navigation tree:

      Domain_Name—Services—JMS—Distributed Destinations

      Note: Distributed destinations for your WebLogic Integration deployment are listed in "JMS Servers and JMS Destinations" on page 2-25.

   b. Create a JMS distributed queue member for each distributed destination.

      For information about creating a distributed queue member, see "JMS Distributed Destination Tasks" in "JMS" the *WebLogic Server Adminstration Console Online Help*, which is available at the following URL:

      `http://edocs.bea.com/wls/docs70/ConsoleHelp/index.html`

Note: For an example of WebLogic Integration distributed destination configuration, see "JMS Servers and JMS Destinations" on page 2-25. You can also view the `JMSDistributedQueue` and `JMSDistributedTopic` elements in the `config.xml` file that was created when you created a domain.

Add a Managed Server in a New Location

To add a managed server to a domain in which the administration server and the clustered managed servers reside on different machines, complete the following steps:

- Step 1. Copy the Contents of Your Preconfigured Domain to the New Location
- Step 2. Modify the Contents of the Directory You Copied
- Step 3. Create a Managed Server
- Step 4. Update the Domain Configuration for the New Managed Server (Optional)

Step 1. Copy the Contents of Your Preconfigured Domain to the New Location

To set up a managed server in a new location, copy the contents of the domain directory you created to the new location, and modify them.

Complete the following procedure:

1. Install WebLogic Integration in the new location.
2. Copy the contents of the domain directory you created (see "Step 2. Create a WebLogic Integration Domain" on page 3-7) to the remote machine. The directory you copy serves as the start location for the managed server.

Note: If you set up a mixed cluster environment (that is, a cluster in which some instances of WebLogic Integration run on Windows systems and others run on UNIX systems), you may encounter a particular problem with carriage return characters. The carriage return character used in scripts run on Windows systems is `^M`. Sometimes these characters remain in a file that is copied from a Windows system to a UNIX system. If the Windows carriage returns persist on a UNIX system, simply open the file and delete the `^M` characters before running your script. You can do this using any text editor, or you can use the `dos2unix` command on Solaris systems. The `dos2unix` utility converts characters in the DOS extended character set to the corresponding ISO standard characters.

If you use FTP to transfer your ASCII files from a Windows system to a UNIX system, you can avoid this problem with carriage return characters by choosing the default ASCII mode.

Step 2. Modify the Contents of the Directory You Copied

Note: The following instructions are based on the assumption that you copied a domain directory named `mydomain` to the following location: *`BEA_HOME/user_projects`*.

To modify the contents of the directory you copied to the new location, delete all files and directories from *`BEA_HOME/user_projects`*, except those listed in the following table.

Files	`startWeblogic.cmd` or `startWebLogic.sh`
	`startManagedWeblogic.cmd` or `startManagedWebLogic.sh`
	`caKeyStore.pks`
	`privateKeyStore.pks`
Directories	`applications`
	`cacerts`
	`certs`
	`keys`
	`wlai`[1]

1. The `wlai` directory is available in the domain directory if you are adding a managed server to a domain in which adapters, application views, and the application integration plug-in are deployed.

Step 3. Create a Managed Server

1. Start the administration server and the Administration Console:

   a. To start the administration server, see "Starting WebLogic Integration" in "Getting Started" in *Starting, Stopping, and Customizing BEA WebLogic Integration*.

   b. To start the console, see "Starting the WebLogic Server Administration Console" in "WebLogic Integration Administration and Design Tools" in *Starting, Stopping, and Customizing BEA WebLogic Integration*.

2. In the Administration Console navigation tree, select Servers.

3. Click Configure a New Server.
4. Enter values in the Name, Listen Address (server instance IP address) fields and, if applicable, in the external DNS name field.

   The value you specify for the external DNS name can be a host name or a virtual host name for a multihomed machine.
5. Select the machine name from the Machine drop-down list.
6. Click Create.
7. Select the Cluster tab.
8. Select the appropriate cluster from the Cluster drop-down list.

Note: For more information about creating and configuring servers, clusters, machines, and domains using the WebLogic Server Administration Console, see the *WebLogic Server Adminstration Console Online Help*, which is available at the following URL:

`http://edocs.bea.com/wls/docs70/ConsoleHelp/index.html`

Step 4. Update the Domain Configuration for the New Managed Server (Optional)

You must add a managed server to the configuration for a domain before you can start that server. You may add a managed server to the domain configuration when you are creating the domain, or you may add it after a domain has been created. Therefore this step is optional.

To determine whether you need to complete it and, if so, how to do so, use the same guidelines and procedures described in “Step 2. Update the Domain Configuration for the New Managed Server (Optional)” on page 3-23.

Step 10. Configure WebLogic Integration for Automatic Restart

Whether WebLogic Integration is deployed in a clustered environment or a nonclustered environment, you can configure your system to automatically restart servers that have shut down because of a system crash, hardware reboot, server failure, and so on. You can do this by configuring the Node Manager in one of two ways:

- Edit the configuration file created when you used the Configuration Wizard to create your domain.

  The `config.xml` file in your domain contains comments to help you quickly identify and edit the elements that must be changed. To configure the Node Manager and modify the configuration file for your domain, complete the following procedure:

  a. In the root directory of the domain you created, open the `config.xml` file in a text editor.

  b. Search for comments labeled with the string NM-OPTION, and follow the instructions included in those comments to configure the Node Manager and SSL.

  c. To configure self-health monitoring (that is, to specify the frequency with which the Node Manager checks a managed server's health), see "Step 4. Configure Self-Health Monitoring" on page 4-9.

  d. To start the Node Manager, see "Step 5. Start the Node Manager" on page 4-10.

- Configure the required components through the WebLogic Server Administration Console, following the procedure described in "Configuring WebLogic Integration for Automatic Restart" on page 4-6.

Step 11. Configure WebLogic Integration for Migration from Failed to Healthy Node

To configure your WebLogic Integration deployment such that it can support migration of resources from a failed node to a healthy node, follow the procedure outlined in "Configuring WebLogic Integration for Migration from Failed to Healthy Node" on page 4-13.

Step 12. Configure WebLogic Integration Security

If you want to configure SSL for your cluster, you can do so by uncommenting predefined sections (one section in each Server element) in the `config.xml` file in your domain.

The `config.xml` file contains comments to help you quickly identify and edit the elements that must be changed. To configure SSL and modify the configuration file for your domain, complete the following procedure:

1. Open the `config.xml` file in the root directory of your WebLogic Integration domain.
2. Search for comments labeled with the string SSL-OPTION, and uncomment the appropriate sections to configure SSL for your domain.

For a domain in which B2B integration functionality is deployed in a multinode cluster, you also need to configure keystores, server certificates, startWeblogic scripts, and so on, for every machine in a cluster.

For information about the tasks you must complete, see:

- "Using the Keystore in a Multinode Cluster" in "Configuring the Keystore" in *Implementing Security with B2B Integration*
- Chapter 5, "Using WebLogic Integration Security."

Warning: If your domain is based on the WLI Domain or EAI Domain template, and if you want to configure keystores for your domain, you must set the Deployed attribute in the WebLogic Integration Application element to *false* before you can configure keystores.

The following listing is an excerpt from a `config.xml` file for a WebLogic Integration domain, which shows the Deployed attribute set to false.

Listing 3-6 Setting the Deployed Attribute Before Configuring Keystores

```
<Application Name="WebLogic Integration" Deployed="false"
Path="C:/bea/weblogic700/integration/lib" TwoPhase="true">
```

Step 13. Start the Servers in the Domain

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

- Before You Start the Servers
- Starting Servers in a Domain for Which the Node Manager Is Not Configured
- Starting Servers in a Domain for Which the Node Manager Is Configured
- Monitoring and Shutting Down Your Servers

Before You Start the Servers

Complete the following procedure before you start the servers in your domain:

1. Make sure that the Deployed attribute is set to `true` for the WebLogic Integration Application element in the domain configuration file.

   If you configured keystores for your domain, you should have set this Deployed attribute to *false*, as described in "Step 12. Configure WebLogic Integration Security" on page 3-30.

   The following listing is an excerpt from a `config.xml` file for a WebLogic Integration domain, which shows the Deployed attribute set to true.

   ```
   <Application Name="WebLogic Integration" Deployed="true"
   Path="C:/bea/weblogic700/integration/lib" TwoPhase="true">
   ```

2. If you specified a server name other than the default when you created your domain (described in "Step 2. Create a WebLogic Integration Domain" on page 3-7), make sure that you now change the name of the following directory in your domain:

   DOMAIN_HOME`/applications/DefaultWebApp_myserver`

   In the preceding path, *DOMAIN_HOME* represents the root directory for your domain. Replace the `myserver` string with the name you specified for your administration server.

Starting Servers in a Domain for Which the Node Manager Is Not Configured

To start servers in a domain for which the Node Manager is not configured, complete the following procedure:

1. Start the administration server by executing the `startWebLogic` command:

   ```
   cd DOMAIN_HOME
   startWeblogic
   ```

   In the preceding line, *DOMAIN_HOME* represents the root directory for your domain.

2. After the administration server is started, start the managed servers in your domain by executing the `startManagedWebLogic` command for each managed server, in turn. In other words, go to the root directory in which you installed each managed server instance, and run the `startManagedWeblogic` command:

   ```
   cd DOMAIN_HOME
   startManagedWeblogic managedserver
   ```

   In the preceding line, *`managedserver`* represents the name of a managed server in your domain.

   As each managed server starts, status messages are displayed in the command window.

Starting Servers in a Domain for Which the Node Manager Is Configured

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

1. Start the administration server by executing the `startWebLogic` command:

   ```
   cd DOMAIN_HOME
   startWeblogic
   ```

   In the preceding line, *`DOMAIN_HOME`* represents the root directory for your domain.

2. Start the managed servers in your domain:

   a. If you have not done so already, start the Node Manager on each machine that hosts managed servers. (See "Step 5. Start the Node Manager" on page 4-10.)

   b. In the Administration Console navigation tree, select the name of each managed server, in turn.

   c. In the main console window, select the Control tab.

   d. 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 Adminstration Console Online Help, which is available from the software and at the following URL:

`http://edocs.bea.com/wls/docs70/ConsoleHelp/index.html`

Monitoring and Shutting Down Your Servers

When startup is complete, you can use the WebLogic Server Administration Console to verify deployments and status.

Then use the WebLogic Server Administration Console to shut down your WebLogic Integration application. It is recommended that you do not close the command window or press Ctrl+c to stop WebLogic Integration. To shut down your application gracefully, run the `stopWebLogic` command, as described in "Stopping WebLogic Integration" in "Getting Started" in *Starting, Stopping, and Customizing BEA WebLogic Integration*.

CHAPTER

4 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

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*, which is available at the following URL:

http://edocs.bea.com/wls/docs70/cluster/index.html

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)
- A shared file system—A shared file system is required for a cluster that uses B2B integration functionality, if you want that cluster to be highly available. We recommend that you use either a Storage Area Network (SAN) or a multiported disk system.
- An Oracle, Microsoft SQL, or Sybase database—You should take advantage of any high availability or failover solutions offered by your database vendor. (See "Recovering a Database" on page 4-22.)
- Persistence mode—Deploy your application with persistence mode turned on (the default setting for WebLogic Integration). Whether your WebLogic Integration application is deployed in a cluster or on a single server, you must run it in persistent mode if you want to be able to recover after a system failure.

A cluster in which B2B integration functionality is used will not work if persistent mode is off.

When you run WebLogic Integration with persistent mode on, the in-memory, dynamic state of objects is saved to persistent storage in the WebLogic Integration repository from where it can be retrieved if necessary. Persistence mode ensures that run-time states can be recovered in the event of an abnormal shutdown or crash.

A 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*, which is available at the following URL:

`http://edocs.bea.com/wls/docs70/cluster/planning.html`

What to Expect from WebLogic Integration Recovery

A highly available deployment has recovery provisions in the event of system failures. You can configure WebLogic Integration for automatic restart or manual migration:

- You can configure WebLogic Integration for automatic restart on a managed server, whether or not that server is in a clustered environment. For information, see "Configuring WebLogic Integration for Automatic Restart" on page 4-6.
- You can configure WebLogic Integration to allow manual migration from a failed node to a healthy node in a clustered environment. For information, see "Configuring WebLogic Integration for Migration from Failed to Healthy Node" on page 4-13.

Note: High availability is not supported for WebLogic Integration applications that are based on the XOCP business protocol; such applications are not recoverable.

When you configure WebLogic Integration appropriately, you can expect the following behavior from your deployment:

- When a server fails, WebLogic Integration reestablishes connections to a live server and then it retries transactions.
- When message delivery fails:
  - In the case of RosettaNet messages, the WebLogic Integration protocol layer does not retry messages; instead it returns HttpStatus code to the workflow layer. RosettaNet workflows are usually designed to handle retries.
  - In the case of ebXML messages, you specify message retries when you specify the ebXML delivery semantics: *once and only once*. Based on the retry value you specify, the WebLogic Integration protocol layer performs message retries for failed ebXML messages.
- WebLogic Integration supports the manual migration of resources from a failed node to a live node in a cluster. For more information, see "Configuring WebLogic Integration for Automatic Restart" on page 4-6 and "Configuring WebLogic Integration for Migration from Failed to Healthy Node" on page 4-13.
- All WebLogic Integration resources that were running before a server crashes will be running again when the server is restarted or fails over.
- If the administration server for a cluster is down, deployment or undeployment requests are interrupted, but managed servers should continue serving requests. You can boot or reboot managed servers using an existing configuration. However, you cannot change configuration for the cluster (for example, add new nodes to the cluster) until the administration server is recovered. For more information, see "Backup and Failover for an Administration Server" on page 4-18.
- If a managed server in a cluster crashes, in-flight requests are interrupted, but other managed servers in the cluster continue serving requests.
- When any one of the managed servers participating in a cluster is down, you cannot deploy or undeploy application integration resources. For example, you cannot deploy application integration adapters when any server in your cluster is down.
- In a clustered environment, it is possible that duplicate B2B messages are sent due to failover and retry attempts. In such cases, a non fatal duplicate message exception is logged, and a 202/200 HTTP status is returned. When they are

received, duplicate messages are not delivered to the workflow or application layer.

- If WebLogic Integration and the database are running on the same machine and the machine is unplugged, you should take steps to recover the database before attempting to recover WebLogic Integration. Under ideal conditions, the database is deployed on a separate machine.
- High availability is supported for ebXML and RosettaNet business protocols. (Applications based on these business protocols are recoverable.)

  WebLogic Integration supports the ebXML Message Service Specification v1.0 and the RosettaNet Implementation Framework v1.1 and v2.0.

  If your WebLogic Integration application includes RosettaNet workflows developed in previous versions of WebLogic Integration, you must make changes to those workflows before running your application on WebLogic Integration 7.0. For information about migrating your workflows, see "Migrating WebLogic Integration 2.1 to WebLogic Integration 7.0" in the *BEA WebLogic Integration Migration Guide*.
- If WebLogic Integration fails while an instance of a workflow is being processed, the workflow is rolled back, and upon recovery, it is restarted from the last quiescent point.
- If one instance of WebLogic Integration sends a message to another instance, but the destination instance has failed, you may see one or more error messages, followed by a stack trace, in the server console. The following are examples of the types of error messages displayed:
  - Not able to send RosettaNet Message
  - Peer Gone Exception
- Automatic restart and recovery are supported for both single-node and clustered deployments of WebLogic Integration. Migration is supported for clustered deployments only.
- Automatic restart and recovery are supported for single-node deployments of WebLogic Integration Business Partner (a midweight trading partner). However, you cannot deploy midweight trading partners in a clustered environment. For information about configuring your WebLogic Integration Business Partner deployment for automatic restart in the case of failure, see "Configuring WebLogic Integration for Automatic Restart" on page 4-6.

Configuring WebLogic Integration for Automatic Restart

Whether WebLogic Integration is deployed in a clustered environment or a nonclustered environment, you can configure your system to automatically restart servers that have shut down because of a system crash, hardware reboot, server failure, and so on.

Note: The procedures in this section address a clustered environment, but you can follow the same procedure to configure a nonclustered environment, that is, one in which you deploy an administration server and a managed server.

Node Manager

The procedures in this section describe how to configure your system to start a managed server when the Node Manager is running on the machine on which the managed server is located. The Node Manager is a Java program provided with WebLogic Server that enables you to perform the following tasks for managed servers:

- Start and stop any remote managed server in a domain
- Automatically restart WebLogic Server instances that have shut down because of a system crash, hardware reboot, server failure, and so on
- Automatically monitor the health of WebLogic Server instances and restart instances that have reached a *failed* health state

For information about the Node Manager, see "Managing Server Availability with Node Manager" in *Creating and Configuring WebLogic Server Domains*, which is available at the following URL:

`http://e-docs.bea.com/wls/docs70/admin_domain/index.html`

Complete the following procedures to configure your WebLogic Integration cluster for automatic restart:

Step 1. Configure Managed Servers for Remote Start

You must first configure each managed server in your cluster so that it can be started from a remote server.

To configure managed servers for remote start, complete the following steps:

1. In the WebLogic Server Administration Console navigation tree, select the managed server for which you want to configure automatic start.
2. Select the Configuration tab and then the Remote Start tab.
3. Enter information in the fields shown on the Remote Start tab. The information required is specific to the remote server. The fields on this tab are described in "Server—Configuration—Remote Start" in the *WebLogic Server Adminstration Console Online Help*, which is available at the following URL:

   `http://e-docs.bea.com/wls/docs70/domain_server_config_server-start.html`

Step 2. Configure SSL for Your Administration Server

Because the administration server communicates with the Node Manager using SSL, you must configure SSL for your administration server. Complete the following steps:

1. Add the following line to the administration server `startWeblogic` command file:

   `-Dweblogic.security.SSL.trustedCAKeyStore=`*`WL_HOME`*`\lib\cacerts`

In the previous line, *WL_HOME* represents the directory where WebLogic Server is installed. For example, if you installed WebLogic Platform in the default directory, *WL_HOME* is `C:\bea70\weblogic700\server`.

2. Copy the `democert.pem`, `demokey.pem` and `ca.pem` files from `BEA_HOME\weblogic700\common\templates\domains\wls.jar` to the root directory for your domain.
3. In the WebLogic Server Administration Console navigation tree, select the administration server.
4. Select the Connections tab and then the SSL tab.
5. Enter information in fields in the SSL tab as described in "Server—Connections—SSL" in the *WebLogic Server Adminstration Console Online Help*, which is available at the following URL:

   `http://e-docs.bea.com/wls/docs70/ConsoleHelp/domain_server_connections_ssl.html`

   The `democert.pem`, `demokey.pem` and `ca.pem` files that you copied to the root directory of your domain in step 2 are sample files provided to get you started. You can use them in the following fields when you configure SSL in the Administration Console: Server Certificate File Name, Server Key File Name, Trusted CA File Name.

Step 3. Configure the Node Manager

To configure the Node Manager for a managed server, you must use the WebLogic Server Administration Console to create a machine, specify attributes for the Node Manager on that machine, and deploy the managed server that you configured for remote start on that machine. Specifically, you must complete the following steps:

1. In the Administration Console navigation tree, select Machines.

   The Machines table is displayed in the right pane, showing all the machines defined in the domain.
2. Click Configure a New Machine (or, if you are configuring a UNIX machine, click Configure a New Unix Machine.

   A dialog box is displayed in the right pane, showing the tabs associated with configuring a new machine.

3. Enter a name for the new machine in the Name attribute field and click Create to create a machine instance with the name you specified.
4. On the Node Manager tab, define Node Manager connection and authentication attributes—the address and port on which the Node Manager listens for connections. The default values for *`address:port`* are `localhost:5555`. Click Apply to implement your changes.
5. On the Servers tab, identify the managed server to reside on this machine (that is, a managed server that you configured for remote start in "Step 1. Configure Managed Servers for Remote Start" on page 4-7).

   You can select an existing server to assign to this machine by selecting the server name in the Available column, and click the appropriate arrow to move the server to the Chosen column.
6. Click Apply to implement your changes.

   The new machine entry now specifies the attributes required for two purposes: to connect to the Node Manager process running on the machine; and to identify which instances of WebLogic Server reside on the machine.

Step 4. Configure Self-Health Monitoring

This step describes how to configure the frequency of your managed server's automated health checks, and the frequency with which the Node Manager checks the servers's health. You can also specify whether the Node Manager automatically stops and restarts the server if the server reaches a *failed* health state.

Complete the following procedure for each managed server:

1. In the WebLogic Server Administration Console navigation tree, select the managed server for which you configured automatic start in "Step 1. Configure Managed Servers for Remote Start" on page 4-7.
2. Select the Configuration tab and then the Health Monitoring tab.
3. Specify the following information:
   - Auto Restart—Select Auto Restart to enable the automatic restarting of this managed server by the Node Manager.

- Auto Kill if Failed—Select Auto Kill if Failed to enable automatic kill of a failed server by the Node Manager.
- Restart Interval—The number of seconds during which Node Manager restarts are counted. This attribute is used with the Max Restarts within Interval attribute to limit attempts to restart this server. The default value is 300 seconds.
- Max Restarts within Interval—The maximum number of times the Node Manager can restart this server within the interval specified by the Restart Interval. The default value is 2 restarts.
- Health Check Interval—Periodicity (in seconds) of the server's health checks. This parameter controls the frequency of the server's self-health monitoring and the Node Manager's health queries.
- Health Check Timeout—Interval (in seconds) the Node Manager should wait before timing out its health query to the server.
- Restart Delay Seconds—Interval (in seconds) the Node Manager should wait before restarting the server. This value is used in cases such as when the OS does not allow listen ports to be reused immediately.

Step 5. Start the Node Manager

You can start the Node Manager manually, by running the `java` command from an operating system prompt, or automatically, by running a script.

Syntax for the Start Node Manager Command

The `java` command syntax for starting the Node Manager is as follows:

```
java [java_property=value ...] -D[nodemanager_property=value]
-D[server_property=value] weblogic.nodemanager.NodeManager
```

Caution: You must start the Node Manager from the same directory in which you start the managed server manually.

In the preceding `java` command line:

- `java_property`—Specifies a direct argument to the `java` executable, such as `-ms`, or `-mx`.

Note: To avoid running out of memory, always specify a minimum heap size of 32 megabytes (`-Xms32m`) for the Node Manager.

- `nodemanager_property`—Defines the behavior of the Node Manager process. Table 4-1 describes valid Node Manager properties.

Table 4-1 Values for nodemanager_property Command-Line Argument

Node Manager Property	Description	Default
`weblogic.nodemanager.certificateFile`	Specifies the path to the certificate file used for SSL authentication.	`./config/democert.pem`
`weblogic.nodemanager.javaHome`	Specifies the Java home directory used by the Node Manager to start managed servers on this machine.	none
`weblogic.nodemanager.keyFile`	The path to the private key file to be used for SSL communication with the Administration Server.	`./config/demokey.pem`
`weblogic.nodemanager.keyPassword`	The password used to access the encrypted private key in the key file.	`password`
`weblogic.ListenAddress`	The address at which the Node Manager listens for connection requests. This argument deprecates `weblogic.nodemanager.listenAddress`.	`localhost`
`weblogic.ListenPort`	The number of the TCP port on which the Node Manager listens for connection requests. This argument deprecates `weblogic.nodemanager.listenPort`.	`5555`
`weblogic.nodemanager.nativeVersionEnabled`	For UNIX systems other than Solaris or HP-UX, set this property to `false` to run the Node Manager in non-native mode.	`true`
`weblogic.nodemanager.reverseDnsEnabled`	Specifies whether entries in the trusted hosts file may contain DNS names (instead of IP addresses).	`false`

Table 4-1 Values for nodemanager_property Command-Line Argument (Continued)

Node Manager Property	Description	Default
`weblogic.nodemanager.savedLogsDirectory`	Specifies the path for the directory in which the Node Manager stores log files. The Node Manager creates a subdirectory in the `savedLogsDirectory` named `NodeManagerLogs`.	`./NodeManagerLogs`
`weblogic.nodemanager.sslHostNameVerificationEnabled`	Determines whether or not the Node Manager performs host name verification.	`false`
`weblogic.nodemanager.startTemplate`	For UNIX systems only, this property specifies the path of a script file used to start managed servers.	`./nodemanager.sh`
`weblogic.nodemanager.trustedHosts`	Specifies the path to the trusted hosts file used by the Node Manager.	`./nodemanager.hosts`
`weblogic.nodemanager.weblogicHome`	Specifies the root directory of the WebLogic Server installation. This directory name is used as the default value of `-Dweblogic.RootDirectory` for servers that do not have a configured root directory.	n/a

- `server_property`—Specifies default values when starting new managed server instances. Table 4-2 describes valid server properties.

Table 4-2 Values for server_property Command Line Argument

Server Property	Description	Default
`bea.home`	Specifies the BEA home directory used by managed servers on the current machine.	Specifies the BEA home directory used by managed servers on the current machine.
`java.security.policy`	Specifies the path to the security policy file that Managed Servers use.	none

Table 4-2 Values for server_property Command Line Argument (Continued)

Server Property	Description	Default
`weblogic.security.SSL.trustedCAKeyStore`	Specifies the path to the KeyStore in which certificates of trusted authorities are contained.	`java.security.keyStore`

The information in the preceding tables, along with details about configuring and running the Node Manager, are available in "Starting Node Manager" in "Managed Server Availability with Node Manager" in *Creating and Configuring WebLogic Server Domains*, which is available at the following URL:

`http://edocs.bea.com/wls/docs70/admin_domain/index.html`

Starting the Node Manager When a Machine Is Booted

In a production environment, the Node Manager should start automatically when a machine is booted. You can ensure that it does so by creating a startup script for UNIX systems, or by setting up the Node Manager as a Windows service for Windows systems. For information about how to perform these tasks, see "Starting Node Manager" in "Managed Server Availability with Node Manager" in *Creating and Configuring WebLogic Server Domains*, which is available at the following URL:

`http://edocs.bea.com/wls/docs70/admin_domain/index.html`

Configuring WebLogic Integration for Migration from Failed to Healthy Node

When a managed server fails and is deemed not to be usable, you can migrate the services from the failed managed server to a healthy node in the cluster. Complete the following procedures to configure your system for a manual migration:

- Step 1. Configure Your Cluster
- Step 2. Configure Migratable Targets for JMS Servers and JTA Recovery Service

For instructions about how to perform the migration when a node in your cluster fails, see "Manual Migration of WebLogic Integration from Failed to Healthy Node" on page 4-19.

Step 1. Configure Your Cluster

Make sure that your WebLogic Integration resources are distributed appropriately and your clustered domain is configured as described in Chapter 3, "Configuring a Clustered Deployment."

Step 2. Configure Migratable Targets for JMS Servers and JTA Recovery Service

To achieve high availability for your WebLogic Integration deployment, you must configure JTA and JMS servers for failover; the process involves configuring migratable targets for JMS servers and the JTA Recovery Service. You can do this by using the WebLogic Server Administration Console or by editing your `config.xml` file appropriately.

Complete the following procedure:

1. Create a migratable target for a cluster:
   a. Select the Servers node in the Administration Console navigation tree.
   b. Select the name of a server that resides in the cluster you want to configure.
   c. Choose Control—Migration Config. in the main console window. A list of servers for which you can select constrained candidate servers for the migratable target is displayed.
   d. In the Available column, select all of the servers capable of hosting migratable services in the cluster. Use the arrow to move these servers to the Chosen column.

      Note: Typically, all the managed servers in the cluster are selected as potential hosts for migratable services.

e. Click Apply to make your changes to the migratable target.

The list of servers that you specified is configured in the `MigratableTarget` element. See the `ConstrainedCandidateServers` attribute for the `MigratableTarget` elements in Listing 4-1. (A domain configuration contains one `MigratableTarget` element for each managed server.)

2. Configure JTA failover:

a. Make sure that all servers in your cluster have access to the transaction log files for a server. In other words, JTA log files should reside on a shared file system.

b. Select the Servers node in the Administration Console navigation tree.

c. Select the name of a server that resides in the cluster you want to configure.

d. Choose Control—JTA Migration Config. to create a migratable target for JTA services. A list of servers that you can select as constrained candidate servers for the migratable target is displayed.

e. In the Available column, select all the servers capable of hosting migratable services in the cluster. Use the arrow to move these servers to the Chosen column.

f. Click Apply to make your changes to the new migratable target.

Note: JTA and JMS service migration is a two-step process. It is recommended that when you migrate WebLogic Integration resources, you first migrate JTA services, and then migrate JMS services. For more information, see "Manual Migration of WebLogic Integration from Failed to Healthy Node" on page 4-19.

For more information about configuring migratable targets, see:

- "Constraining the Servers to Which the Transaction Recovery Service Can Migrate" in "JTA" in *WebLogic Server Adminstration Console Online Help*, which is available at the following URL:

  `http://edocs.bea.com/wls/docs70/ConsoleHelp/jta.html`

- "Server Migration Tasks" in "Servers" in *WebLogic Server Adminstration Console Online Help*, which is available at the following URL

  `http://edocs.bea.com/wls/docs70/ConsoleHelp/servers.html`

Note: Online Help is accessible from the Administration Console, and also at the following URL:

http://edocs.bea.com/wls/docs70/ConsoleHelp/index.html

The following listing, an excerpt from a sample `config.xml` file, shows how to configure migratable targets. It demonstrates the configuration of migratable targets for both JMS servers and the JTA Recovery Service in a clustered WebLogic Integration environment. In this example configuration, the cluster contains two managed servers: `MyServer-1` and `MyServer-2`.

Listing 4-1 Configuration for Migratable Targets

```
<JMSServer Name="WLCJMSServer-MyServer-1"
    Store="JMSWLCStore-MyServer-2" Targets="MyServer-1 (migratable)"
    TemporaryTemplate="TemporaryTemplate">
    <JMSQueue JNDIName="com.bea.b2b.OutboundQueue-MyServer-1"
        Name="B2bOutboundQueue-MyServer-2"/>
    <JMSQueue ...
     :
</JMSServer>

<JMSServer Name="WLCJMSServer-MyServer-2"
    Store="JMSWLCStore-MyServer-2" Targets="MyServer-2 (migratable)"
    TemporaryTemplate="TemporaryTemplate">
    <JMSQueue JNDIName="com.bea.b2b.OutboundQueue-MyServer-2"
        Name="B2bOutboundQueue-MyServer-2"/>
    <JMSQueue ...
     :
</JMSServer>
...

<MigratableTarget Cluster="MyCluster"
    ConstrainedCandidateServers="MyServer-1,MyServer-2"
    Name="MyServer-1 (migratable)"
    Notes="This is a system generated default migratable target for a server.
    Do not delete manually."
    UserPreferredServer="MyServer-1"/>

<MigratableTarget Cluster="MyCluster"
    ConstrainedCandidateServers="MyServer-1,MyServer-2"
    Name="MyServer-2 (migratable)"
    Notes="This is a system generated default migratable target for a server.
    Do not delete manually."
    UserPreferredServer="MyServer-2"/>
```

```
...
<Server Cluster="MyCluster" JTARecoveryService="MyServer-1"
    ListenAddress="localhost" ListenPort="7901" Name="MyServer-1"
    ServerVersion="7.0.0.0">
    <COM Name="MyServer-1"/><ExecuteQueue Name="default" ThreadCount="15"/>
    <IIOP Name="MyServer-1"/>
    <JTAMigratableTarget Cluster="MyCluster"
    ConstrainedCandidateServers="MyServer-1,MyServer-2 Name="MyServer-1"
    UserPreferredServer="MyServer-1"/>

</Server>

<Server Cluster="MyCluster" JTARecoveryService="MyServer-2"
    ListenAddress="localhost" ListenPort="7901" Name="MyServer-2"
    ServerVersion="7.0.0.0">
    <COM Name="MyServer-2"/><ExecuteQueue Name="default" ThreadCount="15"/>
    <IIOP Name="MyServer-2"/>
    <JTAMigratableTarget Cluster="MyCluster"
    ConstrainedCandidateServers="MyServer-1,MyServer-2 Name="MyServer-2"
    UserPreferredServer="MyServer-2"/>

</Server>
```

Note the following XML elements in the preceding listing:

- JMS Server—The Target attribute for the JMS server must be the migratable target name for the server. A migratable target is created, by default, for each managed server. (For details, see the following item in this list.)
- MigratableTarget—The migratable target specification for `MyServer-2` in `MyCluster`. As noted in the listing, the migratable target is a system-generated default migratable target for a server. One such target is created for each managed server in a cluster.

  The `MigratableTarget` element also contains the comma-separated list of servers for `ConstrainedCandidateServers`. The servers in this list are those that you have specified as capable of acting as JMS server backups. Note that you must include the `UserPreferredServer` in the list of `ConstrainedCandidateServers`; the WebLogic Server Administration Console enforces this rule.
- Server—The Server elements include the specification of migratable targets for `JTARecoveryService`.

Failover and Recovery

This section describes how WebLogic Integration failover and recovery works in specific scenarios. It contains the following topics:

Backup and Failover for an Administration Server

To provide for quick failover in case of an administration server crash or other failure, you may wish to create another instance of the administration server on a different machine that is ready to use if the original server fails.

Because the administration server uses the configuration file (`config.xml`), security files, and application files to administer the domain, we recommend that you at least keep an archived copy of these files, so that in case of a failure of the administration server you can safely restart the administration server on another machine without interrupting the functioning of the managed servers.

When the administration server for a cluster crashes, the managed servers continue serving requests. However, you cannot change configuration for the cluster or perform new deployment activities until the administration server is recovered. For example, if the administration server for a cluster is not running, you cannot add new nodes to the cluster, deploy new application views, or undeploy the connection factories associated with those application views.

The WebLogic Integration B2B Console is deployed only on the administration server. It is never deployed on the managed servers in a cluster. Therefore B2B integration management and monitoring functions are unavailable when the administration server is down. For example, you cannot add, delete, or modify trading partner information until the administration server is recovered.

If the managed servers are running but the administration server is stopped, you can recover management of the domain without the need to stop and restart the managed servers.

For instructions to restart your administration server when managed servers are running, see "Starting and Stopping WebLogic Servers" in the *BEA WebLogic Server Administration Guide*, which is available at the following URL:

`http://e-docs.bea.com/wls/docs70/adminguide/startstop.html`

Manual Migration of WebLogic Integration from Failed to Healthy Node

This section describes a controlled fail over. That is, source and destination servers are not serving any requests when you migrate services from a failed node to a healthy node in your cluster.

Before attempting to migrate WebLogic Integration from a failed node to a healthy node:

- You should have configured your application (before starting your cluster) as described in "Configuring WebLogic Integration for Migration from Failed to Healthy Node" on page 4-13.
- Ensure that your source server is not running. If the source server is not down, but only unavailable because of network problems, the services will be copied to the destination server without being removed from the source server, resulting in two simultaneous running versions of the same services, which could cause corruption of the transaction log or of JMS messages.

Note: JTA and JMS service migration is a two-step process. When you migrate WebLogic Integration resources, you should first migrate JTA services, and then migrate JMS services.

You can migrate WebLogic Integration using one of the following methods:

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

Using the weblogic.Admin Command-Line Utility

Use the following command line (`weblogic.Admin` with the `MIGRATE` command) to migrate a JMS service or a JTA service to a targeted server within the cluster:

```
java weblogic.Admin [-url http://hostname:port]
[-username username]
[-password password]
MIGRATE -jta -migratabletarget (migratabletarget_name|servername)
-destination servername [-sourcedown] [-destinationdown]
```

In the previous command line:

- `-url`—Specifies the URL of the administration server, including the number of the TCP port at which WebLogic Server is listening for client requests. The format is *`hostname:port`*. The default is `localhost:7001`.
- `-jta`—Specifies that the migration is a migration of JTA services. If `-jta` is not specified, the migration is assumed to be a migration of JMS services.
- `-migratabletarget`—Names a configuration file identified with the server from which services will migrate. For each server, WebLogic Server automatically creates a migratable target file. (The file is named *`servername`*`_migratable` for JMS and `servername` for JTA.) This migratable target file is a configuration file that specifies the preferred servers for JMS and JTA services.
- `-destination`—The name of the server to which the services will migrate.
- `-sourcedown`—Specifies that the source server is down.

  Warning: As mentioned previously in this section, it is important to ensure that your source server is down when you invoke `weblogic.Admin` with the `MIGRATE` command.
- `-destinationdown`—Use this option if the destination server is down when you do the migration.

For more information about the `weblogic.Admin` command-line tool, see "WebLogic Server Command-Line Interface Reference" in *BEA WebLogic Server Administration Guide*, which is available at the following URL:

`http://e-docs.bea.com/wls/docs70/adminguide/cli.html`

Using the WebLogic Server Administration Console

As an alternative to the `weblogic.Admin` command-line tool, you can use the WebLogic Server Administration Console to migrate a JTS service or a JMS service to a targeted server within the cluster:

1. Select the Servers node in the Administration Console navigation tree.
2. Select the name of a server in your cluster.
3. Select the Migrate tab appropriate for the services you are migrating. Note that JTA and JMS service migration is a two-step process. When you migrate WebLogic Integration resources, you should first migrate JTA services, and then migrate JMS services. The Migrate tab you select is determined by the type of services you are migrating:
   - When you migrate JTA services, select the Control tab, followed by the JTA Migrate tab.
   - When you migrate JMS services, select the Control tab, then the Migrate tab.

   Warning: Make sure that your source server is down. Failover for WebLogic Integration is supported only when the source server is down.
4. Select a server from the Destination Server migratable target list.
5. Click Migrate.

   Services running on the server you selected in step 2 are migrated to the destination server you selected.

   The services that are migrated depend on the selection you made in step 3. That is, if you selected the JTA Migrate tab, only the JTA service is migrated to the selected server. If you selected the Migrate tab, only JMS services are migrated to the selected server.

For more information about how to use the Administration Console to migrate JMS or JTA services to a targeted server within the cluster, see "Migrating Services to a New Server" in "Servers" in *WebLogic Server Adminstration Console Online Help*.

Recovering a Database

WebLogic Integration does not attempt to recover a crashed database. In the event of a database crash or database shutdown, it may be necessary to restart WebLogic Integration.

For example, if WebLogic Integration and the database are running on the same machine and the machine is unplugged, you should take steps to recover the database before attempting to recover WebLogic Integration.

Recovering JMS Stores

There is no migration of JMS stores after a server crash. WebLogic Integration uses JDBC for JMS stores. That is, it uses JDBC to access JMS JDBC Stores, which can be on another server. WebLogic Integration uses the same database for all nodes in a cluster. If you plan to use separate database instances for each node in your cluster, you should take advantage of any high availability or failover solutions offered by your database vendor. For example, you could use warm database standby in the event of database crash.

CHAPTER

5 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, see *Introducing WebLogic Platform 7.0 Security*, which is available at the following URL:

`http://edocs.bea.com/platform/docs70/secintro/index.html`

This document provides an overview of the security features of the entire WebLogic Platform, and provides important notes about managing security when using WebLogic Integration with other WebLogic Platform components.

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. Therefore, 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:

- WebLogic Integration system user, `wlisystem`
- Users of the BPM engine and WebLogic Integration Studio, and the groups to which they belong
- 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
- Application views

As the security manager 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 BEA Configuration Wizard, the domain is configured, by default, to use compatibility security. Compatibility security enables a domain to do the following:

- Use existing stores of users and groups to authenticate WebLogic Server principals
- Use access control lists (ACLs) to protect WebLogic Integration resources

By default, all WebLogic Integration users, groups, and ACLs are stored in a security realm known as the compatibility realm.

Note: A typical installation of WebLogic Integration includes WebLogic Server and WebLogic Workshop components. By default, the Configuration Wizard configures WebLogic Integration security to use compatibility security and allocates the WebLogic Server 6.x File realm for storing users and groups. The File realm is used in all WebLogic Integration samples. The WebLogic Server and WebLogic Workshop samples are based on a security configuration that is, in turn, based on an embedded LDAP server, which WebLogic Integration does not support in this release. Therefore, the samples delivered with WebLogic Server and WebLogic Workshop may not work with the default configuration for the WebLogic Integration samples.

For more information about the BEA Configuration Wizard, see *Using the Configuration Wizard*, which is available at the following URL:

`http://edocs.bea.com/platform/docs70/confgwiz/index.html`

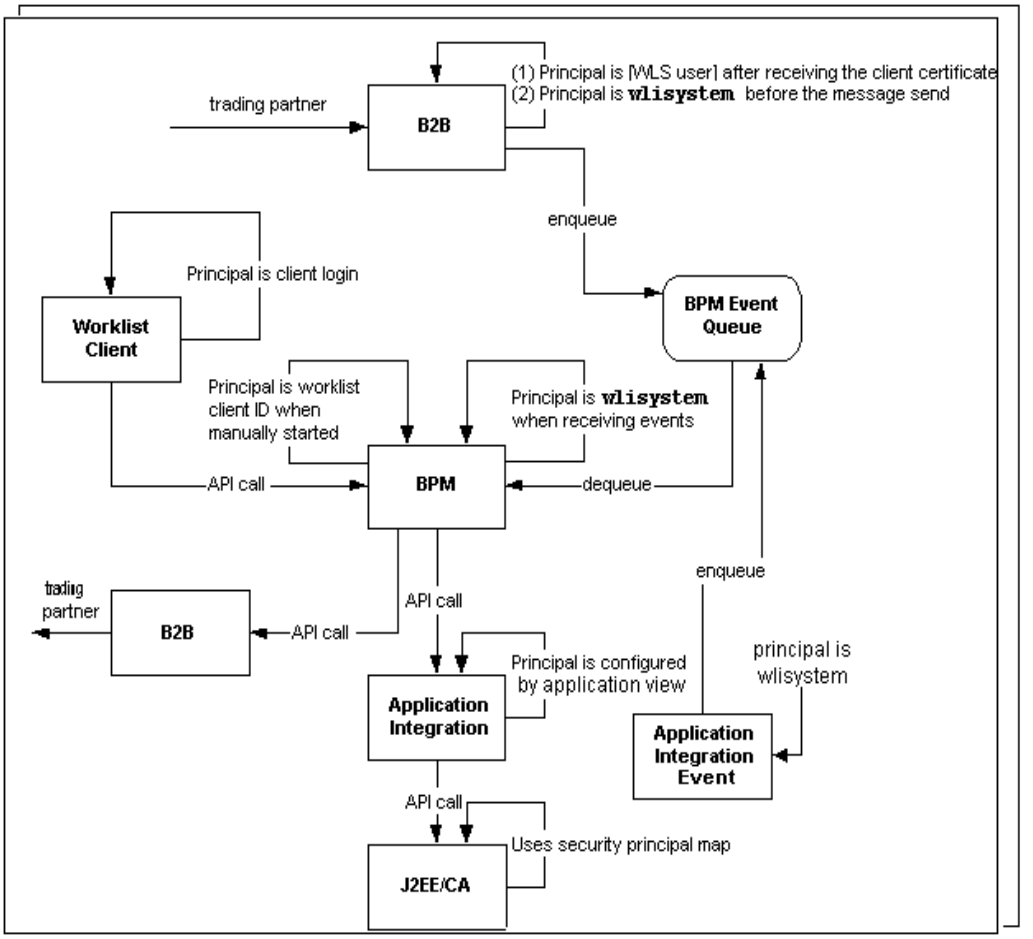
WebLogic Server Security Principals and Resources Used in WebLogic Integration

When you create a WebLogic Integration domain via the Configuration Wizard, the following WebLogic Server principals and resources are predefined:

- `wlisystem`—WebLogic Integration system user that functions on behalf of a trading partner after the trading partner is authenticated and authorized to use WebLogic Integration resources.
- `wlpiUsers`—Business Process Management users group that has access to BPM resources, such as the WebLogic Integration Studio and the BPM engine.
- Set of BPM groups—Each member of the `wlpiUsers` group is a member of one or more of these groups. Membership in a group determines the set of tasks that can be performed by a principal. Each group represents an ACL.
- `wliPool`—JDBC connection pool for the WebLogic Integration repository.
- `ServletFilter`—Resource that is used transparently by trading partners to access B2B resources.

The following diagram provides an overview of the WebLogic Server security principals used in WebLogic Integration.

Figure 5-1 WebLogic Server Security Principals Used in WebLogic Integration



For example, during the course of a B2B-based business operation, the WebLogic Server principals may function as follows:

- When started via an event (XML, application integration, or B2B integration), a workflow runs as `wlisystem`.
- When started via a manual task start, a workflow runs as the security principal associated with the Worklist client.

- An application integration service runs as the security principal configured for the application view.
- When a message is received from a trading partner, the B2B engine momentarily switches to the WebLogic Server principal associated with the client-side certificate for the trading partner. Then it switches to the `wlisystem` principal before sending the message to the BPM event queue. For more information about B2B integration security, see *Implementing Security with B2B Integration*.
- When the J2EE-CA adapter receives a request, it maps the caller's security principal to one that is appropriate for the EIS system. For more information, see "Deprecated Security Principal Map Mechanism" in *Programming the WebLogic J2EE Connector Architecture* in the WebLogic Server documentation set, at the following URL:

  `http://edocs.bea.com/wls/docs70/jconnector/security.html`

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 or NonWebLogic Server Proxy Server

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 interenterprise commerce, using the B2B 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 B2B 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 "Configuring Security" in *Implementing Security with B2B 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.

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 B2B collaborations, you need to configure the following:

- SSL for each machine in your WebLogic Integration domain. For information about how to do this, see "Configuring the SSL Protocol and Mutual Authentication" in "Configuring Security" in *Implementing Security with B2B Integration*.
- Set of digital certificates and private keys for each trading partner
- Server certificate for each machine in the WebLogic Integration domain
- Certificate for the root Certificate Authority (CA)

  For more information about configuring certificates, see "Configuring Security" in *Implementing Security with B2B Integration*.

Not required by SSL, but strongly recommended, is the creation and use of keystores for storing all the certificates and keys used in your WebLogic Integration domain. WebLogic Server provides a utility called the *WebLogic Keystore provider* based on the reference Keystore implementation supplied by Sun Microsystems in the Java Development Kit.

The WebLogic Keystore provider is based on the standard *JKS* keystore type, which implements the keystore as a file. For this release of WebLogic Server, JKS is the only keystore provider available. A keystore configured with the WebLogic Keystore provides protects each private key with an individual password. Two keystore files are associated with the WebLogic Keystore provider: One holds the CA certificates used by SSL to verify client certificates; the other holds users' private keys. WebLogic Server retrieves a private key from this keystore to initialize SSL.

For more information about setting up keystores for your WebLogic Integration domain, see "Configuring the Keystore" in *Implementing Security with B2B Integration*.

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 WebLogic Integration B2B to Use an Outbound HTTP Proxy Server" in "Configuring Security" in *Implementing Security with B2B 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, consider the following:

- 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 Authority" in "Introducing WebLogic Integration B2B Security" in *Implementing Security with B2B Integration*.)
- If the local trading partner site uses a Web server configured with a WebLogic proxy plug-in, then you can specify the trading partner transport URI endpoints in the usual manner.
- If the remote trading partner is also using WebLogic Integration, but is using a proxy server other than the WebLogic proxy server, then it is likely that the remote site is configured with the WebLogic proxy plug-in. When you are configuring a remote trading partner under these circumstances, you must specify the host and port of the trading partner's proxy server as the transport URI endpoints. The WebLogic proxy plug-in performs the necessary URL transformations to business messages received for that remote trading partner.

Using a Firewall

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

Setting Up a Secure Deployment

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

- Step 1: Create the Domain
- Step 2: Configure WebLogic Server Security
- Step 3: Configure BPM Security

- Step 4: Configure B2B Integration Security
- Step 5: Configure Application Integration Security

Step 1: Create the Domain

We recommend that you use the BEA Configuration Wizard to create the WebLogic Integration domain for which you want to configure security. To create a WebLogic Integration domain, complete the following steps:

1. Start the Configuration Wizard, as described in *Using the Configuration Wizard*, available at the following URL:

   `http://edocs.bea.com/platform/docs70/confgwiz/index.html`

2. Complete the configuration of the WebLogic Integration domain, which can be any of the following:

   - WebLogic Integration BPM domain
   - WebLogic Integration EAI domain
   - WebLogic Integration domain

Note: Make sure you use a WebLogic Integration template when creating the new domain; do not use a WebLogic Server or a WebLogic Portal template. By specifying a WebLogic Integration template, you can make sure that the domain created in this step is based on the WebLogic Server 6.x security realm in compatibility mode. The new WebLogic Server 7.0 realm, based on LDAP, is not supported with WebLogic Integration. If you create a new domain by selecting a WebLogic Server template, the new domain uses the new WebLogic Server 7.0 security realm, which is based on LDAP.

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:

   - 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 requirements applies to the server certificate for *any* trading partner in any collaboration agreement configured locally. 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.

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

     "About Digital Certificates" on page 4-7 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 root CA keystore to store trusted CA certificates, be sure to import the whole chain in to the root CA keystore.

3. Configure the WebLogic Keystore provider. WebLogic Server 7.0 supports keystore functionality. For complete details on creating keystores and configuring the WebLogic Keystore provider, see "Configuring the Keystore" in *Implementing Security with B2B Integration*.

   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 of 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 BPM Security

The security model provided by WebLogic Integration for business process management (BPM) functions comprises the following entities:

- Users and groups—A user is an individual who performs a certain task, such as programming or sales. A group is a collection of one or more users or groups that perform the same task. For example, Group A might represent a collection of users who are programmers; Group B, a collection of sales people.

  Within a security realm, the administrator (represented by the principal `wlisystem`) can specify the levels of access given to users and groups who want to use workflows and other resources. Users and groups are maintained in a WebLogic Server security realm. You can define groups and add users to those groups in the WebLogic Integration Studio. Each user you add is automatically and simultaneously added to the list of WebLogic Server users.

- Organizations—Entities that can represent different business organizations, geographical locations, or any other distinguishing items that are relevant to the particular business of an enterprise.

  Organizations are BPM-specific entities that exist outside the WebLogic Server security realm.

- Roles—Common areas of responsibility, ability, or authorization level that is shared by a group of individuals. A role may belong to one organization, but you can use the same name in multiple organizations.

  Roles map to WebLogic Server groups.

The task of configuring BPM security is basically one of defining users, groups, roles, organizations, and permission levels. Because you can define organizations and roles, you have a great deal of flexibility in organizing the users and groups that access BPM resources. The Studio provides tools that allow you to create and modify users, groups, roles, and organizations. The Studio also provides a method for managing permissions for users, groups, and roles in minute detail.

For more information about BPM security, see the following topics:

- "About Security Realms" in "Administering Data" in *Using the WebLogic Integration Studio*
- "Understanding the BPM Security Model" in "Customizing WebLogic Integration" in *Starting, Stopping, and Customizing BEA WebLogic Integration*. See especially the topic, "Configuring a Custom Security Realm."
- "Configuring the Security Realms" in *Programming BPM Client Applications*

Step 4: Configure B2B 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 B2B security, you must perform the following tasks:

- Obtain the certificates and keys required for conducting B2B collaborations. Required certificates include those for the root CA, 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.
- Create keystores for the certificate keys used in a WebLogic Integration environment, and register them with the WebLogic Keystore provider.
- Configure local trading partners.
- Configure remote trading partners.
- Implement nonrepudiation services, if desired.
- Implement the security requirements for the business protocols used.

The sections that follow provide recommendations and considerations for each of these tasks.

Obtaining Certificates

Before you begin configuring WebLogic Integration security, particularly if you plan to conduct B2B exchanges, make sure you have the following certificates and keys:

- Server certificates and keys for each instance of WebLogic Server used in the B2B exchanges. This requirement also applies to remote trading partners: for each remote trading partner you configure, you must have that trading partner's server certificate. These certificates are required by SSL.
- Root Certificate Authority for each certificate used in the environment.

  The root CA directory should contain *only* the root CA certificates for each trading partner's client, encryption, and signature certificate. The root CA directory must *not* contain the CA certificates for the server certificates. (The server certificates are configured in the domain's `config.xml` file.)
- Client certificates for each trading partner, whether local or remote. For local trading partners, you must also obtain the location of the client certificate's private key.
- Encryption and signature certificates for all trading partners. For local trading partners, you must also obtain the location of each certificate's private key.

Creating the Keystores

When you set up a WebLogic Integration domain for B2B collaborations, you must configure the WebLogic Keystore provider to create the following keystores:

- Private keystore

  Stores the local trading partners' private keys and certificates, such as the client, server, signature, and encryption certificates typically required for B2B collaborations. WebLogic Server retrieves a private key from this keystore to initialize SSL.
- Root CA keystore

  Stores the certificates of all the trusted certificate authorities (CAs). The WebLogic Keystore provider creates a trusted CA keystore that WebLogic Server uses, by default, to locate the trusted CAs used by SSL to verify client certificates.

You can use the JavaSoft JDK `keytool` utility or the WebLogic Server `ImportPrivateKey` utility to create each keystore and to add private keys to it. If neither keystore exists, it is created the first time you use either of these utilities to add a private key.

After you create the keystores and populate them with initial sets of keys, register them with the WebLogic Keystore provider, as described in "Step 2: Configure WebLogic Server Security" on page 4-11.

Configuring Local Trading Partners

Local trading partners send messages to remote trading partners using either HTTP or HTTPS. If you are using SSL in your B2B collaborations, which we strongly recommend, you need to configure the client certificate and key for each local trading partner.

Note the following about client certificates and keys for local trading partners:

- Both plain and password-protected keys are supported. The PEM, DER, PKCS8 formats are supported.
- PEM or DER encoded X509 V1 or V3 certificates are supported.
- The password for the private key is case-sensitive and is specified in the command line of the `startWeblogic` script in a Java system property.

Note the following about signature and encryption certificates and keys for local trading partners:

- Only password-protected private keys, using the PKCS8, format are supported.
- DER encoded certificates is recommended.
- You must copy the root CA certificate to the location pointed to by the CA certificate directory.
- The private key password is case-sensitive and is specified in the command line of the `startWeblogic` script in a Java system property.

Configuring Remote Trading Partners

If you are using SSL, as with local trading partners, you need to configure the client certificate and key for each remote trading partner.

Note the following about client certificates and keys for remote trading partners:

- The remote trading partner uses a client certificate to establish a two-way SSL connection with WebLogic Server in your domain. Make sure that this certificate

matches the one specified for that trading partner in the WebLogic Integration repository.

- If the remote trading partner is also uses WebLogic Server, this information is stored in the repository as that trading partner's client certificate.
- If the remote trading partner is not based on WebLogic Integration, obtain a valid client certificate form that trading partner. (In the case of WebLogic Integration - Business Connect, you can use that software's administration tool to export a company's certificate to a file, and specify that file's location in WebLogic Integration.)

Note the following about signature and encryption certificates and keys for remote trading partners:

- Make sure you obtain both the signature and encryption certificates before configuring these certificates in the WebLogic Integration repository. Not all these certificates need to be same; however, one certificate may be used for both encryption and signatures.
- You must also obtain the CA certificates and add them to the root CA keystore (or directory if you are not using keystores). Self-signed certificates should also be added, if used.

Note the following about the remote trading partner's server certificate:

- You need to obtain the HTTP(S) server certificate of the remote trading partner's site. This is the server to which you establish the SSL connection.
- If the remote trading partner's site is also based on WebLogic Integration, and the remote site's Web server is WebLogic Server, use the certificate of that Web server as the trading partner's server certificate.

Implementing the Security Requirements for Business Protocols

Note that the business protocol with which a Collaboration Agreement is configured may have additional specific security requirements.

The following table lists additional sources of information about various business protocols.

Table 5-1 More Information About Business Protocols Used in B2B

For information about . . .	See this section . . .	In this document . . .
RosettaNet security	"Configuring RosettaNet Security" in "Introduction"	*Implementing RosettaNet for B2B Integration*
cXML security	"Security" in "cXML Administration"	*Implementing cXML for B2B Integration*
ebXML security	"Configuring Security" in "Administering ebXML"	*Implementing ebXML for B2B Integration*

Step 5: 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 Application Integration*.
- 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 "Deploying an Application View" in "Steps for Defining an Application View" in "Defining an Application View" in *Using Application Integration*.

CHAPTER

6 Tuning Performance

The following sections describe how to tune the performance of your WebLogic Integration deployment:

- Tuning WebLogic Integration Performance
- Monitoring and Tuning Run-Time Performance
- Tuning Hardware, Operating System, and Network Resources
- Tuning Databases

Tuning WebLogic Integration Performance

The following sections describe how to tune WebLogic Integration performance:

- Primary Tuning Resources
- Tuning WebLogic Server Performance
- Monitoring and Tuning the Java Virtual Machine (JVM)

Primary Tuning Resources

This section describes the primary WebLogic Integration resources that you can tune to manage the work that a server performs:

- For BPM, the primary resource to tune for event-driven workflows is the event listener message-driven bean.

- For application integration, tuning depends on the type of processing:
  - For synchronous service invocations, the primary resource is the application view bean.
  - For asynchronous service invocations, the primary resource is the thread pool size of the asynchronous request processor.
  - Event adapters usually do not require tuning.

  In addition, the J2EE-CA resource pool size should be set for each adapter. For information about how to tune an adapter, see the documentation for the adapter.
- For B2B integration, there are no primary resources that can be tuned.

All other WebLogic Integration resources should be changed only to support these primary resources.

Tuning WebLogic Server Performance

The following sections describe how to configure WebLogic Server resources for a WebLogic Integration deployment:

- Configuring EJB Pool and Cache Sizes
- Configuring JDBC Connection Pool Sizes
- Configuring the Execution Thread Pool
- Configuring Resource Connection Pools for J2EE Connector Architecture Adapters
- Configuring Large Message Support for B2B

For general information about tuning WebLogic Server performance, see *BEA WebLogic Server Performance and Tuning* at the following URL:

`http://edocs.bea.com/wls/docs70/perform/index.html`

Configuring EJB Pool and Cache Sizes

You can tune WebLogic Integration performance by configuring EJB pool sizes and cache sizes: start with the default settings and change them as needed. From a performance standpoint, an overly large pool or cache size is generally better than an overly small one. For more information about configuring these settings, see "Deploying EJBs to WebLogic Server" in *Programming WebLogic EJB* at the following URL:

`http://edocs.bea.com/wls/docs70/ejb/index.html`

Configuring BPM Event Listener Message-Driven Beans Pool Size

The `wlpi-mdb-ejb.jar` file contains the pool of event listener message-driven beans that pull events off the event queue. The pool size setting controls the number of workflows executed in the WebLogic Integration system, based on incoming events. Like message-driven beans for the event and validating event queues, time event listeners are deployed to the cluster in the `wlpi-mdb-ejb.jar` file. These message-driven beans pull work from `com.bea.wli.bpm.TimerQueue`.

You can access the default EventListener and TimeListener pool sizes by selecting Edit EJB Descriptor for the WLI-BPM Event Processor EJB (`wlpi-mdb-ejb.jar`) in the Administration Console. For example, the default setting for the Event Listener message-driven beans is 10 (5 unordered listeners plus 5 ordered listeners).

If you configure a custom JMS queue for your system, use the MDB Generator utility to set the pool size and associated queue, as described in "Configuring a Custom Java Message Service Queue" in "Customizing WebLogic Integration" in *Starting, Stopping, and Customizing BEA WebLogic Integration*.

We recommend starting with 20 beans and monitoring your system to determine whether you need more. For more information, see "Do You Have Enough Message-Driven Beans?" on page 6-20.

Configuring Other EJB Pool and Cache Sizes

The following cache and pool size settings are important to consider when you tune your system. These parameters can be tuned for each node in your WebLogic Integration cluster:

Cache size for BPM workflow processor beans

BPM workflow processor beans are described in "Workflow Processor Beans" on page 1-11.

This cache size should equal or exceed the size of the BPM event listener message-driven bean pool. It should also accommodate the anticipated workload from subworkflows or Worklist clients. The default setting is 100. To access the `WorkflowProcessor` EJB and its `Max Beans In Cache` setting, select Edit EJB Descriptor for the WLI-BPM Server EJB (`wlpi-ejb.jar`) in the WebLogic Server Administration Console.

Cache size of the BPM template entity beans

BPM template entity beans are described in "Template Beans" on page 1-12.

This cache size should equal or exceed the number of unique templates concurrently processed in the WebLogic Integration system. The default setting is 100. To access the `TemplateDefinitionRO` EJB and its `Max Beans In Cache` setting, select Edit EJB Descriptor for the WLI-BPM Server EJB (`wlpi-ejb.jar`) in the WebLogic Server Administration Console.

Cache size of the BPM instance entity beans

BPM instance entity beans are described in "Instance Beans" on page 1-12.

This setting should equal or exceed the number of workflow instance processors. The default setting is 100. To access the `WorkflowInstance` EJB and its `Max Beans In Cache` setting, select Edit EJB Descriptor for the WLI-BPM Server EJB (`wlpi-ejb.jar`) in the WebLogic Server Administration Console.

Pool size for the Application View stateless session bean

Application View stateless session beans are described in "Application Integration Resources" on page 1-15.

The default setting for the `Max Beans In Free Pool` is 200, which is sufficient for most deployments.To access the `com.bea.wlai.client.ApplicationView` EJB and its Pool setting, select

Edit EJB Descriptor for the WLI-AI Server EJB (`wlai-server-ejb.jar`) in the WebLogic Server Administration Console.

Cache size for the Application View stateful session bean

The default setting for `Max Beans In Cache` is 1000. To access the `com.bea.wlai.client.StatefulApplicationView` EJB and its Stateful Session Cache setting, select Edit EJB Descriptor for the WLI-AI Server EJB (`wlai-server-ejb.jar`) in the WebLogic Server Administration Console.

Pool size for the asynchronous service processor message-driven bean

Asynchronous service processor message-driven beans are described in "Application Integration Resources" on page 1-15.

The default setting for the `Max Beans In Free Pool` is 1000. To access the WLI-AI Async Processor message-driven bean and its Pool setting, select Edit EJB Descriptor for the WLI-AI Async Processor EJB (`wlai-asyncprocessor-ejb.jar`) in the Administration Console.

Pool size for the event processor message-driven bean

Event processor message-driven beans are described in "Application Integration Resources" on page 1-15.

The default setting for the `Max Beans In Free Pool` is 1000. To access the WLI-AI Event Processor message-driven bean and its Pool setting, select Edit EJB Descriptor for the WLI-AI Event Processor (`wlai-eventprocessor-ejb.jar`) in the Administration Console.

Configuring JDBC Connection Pool Sizes

You can tune WebLogic Integration performance by configuring the size of JDBC connection pools. For an introduction, see "JDBC Connection Pools" on page 1-7.

To determine the necessary size of a JDBC connection pool on each node in a WebLogic Integration cluster, calculate the number of required connections per server, based on the guidelines in the following table.

Table 6-1 Calculating Connections for the JDBC Connection Pool

For this resource . . .	Calculate the required number of JDBC connections as follows . . .
BPM event listener message-driven bean pool size (unordered beans + all ordered beans)	Multiply the event listener message-driven bean pool size by 2. For example, if the event listener message-driven bean pool size is 10, you need to add 20 connections to the JDBC connection pool. Event listeners always use at least one—and possibly two—JDBC connections. Multiplying by a factor of 2 accounts for a worst-case scenario, so you can probably use a smaller size if necessary. **Note:** If you run workflow processors from Worklist clients, you need to add more connections.
B2B integration	Add 10 connections to the JDBC connection pool.
Application integration	Add 1 connection for each application view bean (the default is 5) and add 1 connection for each asynchronous request processor listener (the default is 2).
Application integration adapters	Add any connections needed for adapters (event adapters and service adapters). For example, for the DBMS adapter, add one connector for each resource in the J2EE-CA resource connector pool.

After calculating the number of connections required for each resource, calculate the total needed for all resources, and then configure the JDBC connection pool for each node in the cluster, using this total.

For best performance, set the initial capacity and the maximum capacity to the same value.

You can find information on monitoring JDBC connections in "Do You Have Enough JDBC Connections?"

For more information about JDBC connection pools, see the following sections:

- "Tuning JDBC Connection Pool Size" in "Tuning WebLogic Server" in *BEA WebLogic Server Performance and Tuning* at the following URL:

  http://edocs.bea.com/wls/docs70/perform/WLSTuning.html

- "Managing JDBC Connectivity" in the *BEA WebLogic Server Administration Guide* at the following URL:

  http://edocs.bea.com/wls/docs70/adminguide/jdbc.html

Configuring the Execution Thread Pool

You can tune WebLogic Integration performance by configuring the execution thread pool, which is described in "Execution Thread Pool" on page 1-8. For each node in a WebLogic Integration cluster, calculate the number of execution threads based on the guidelines described in the following table.

Table 6-2 Calculating the Number of Execution Threads

For this resource . . .	Calculate the required number of execution threads as follows . . .
BPM	For BPM overhead, add 1 thread.
BPM event listener message-driven beans	For each event listener message-driven bean, add 1 thread.
Concurrent Worklist client requests	For each anticipated simultaneous Worklist client request, add 1 thread.
B2B integration	Add 2 threads for every RosettaNet or ebXML message-driven bean.
Application integration	Add 5 threads for application integration overhead.
Application integration adapters	For each adapter, add 3 threads.
Applications	Add any execution threads required for application use.

After calculating the number of threads required for each resource, calculate the total needed for all resources, and then configure the thread pool size for each server, using this total.

For instructions on how to configure the thread pool size using the WebLogic Server Administration Console, see "Server Configuration Tasks" in "Servers" in *Adminstration Console Online Help* at the following URL:

`http://edocs.bea.com/wls/docs70/ConsoleHelp/servers.html`

For information about monitoring threads on your system, see "Do You Have Enough Threads?" on page 6-13.

Configuring Resource Connection Pools for J2EE Connector Architecture Adapters

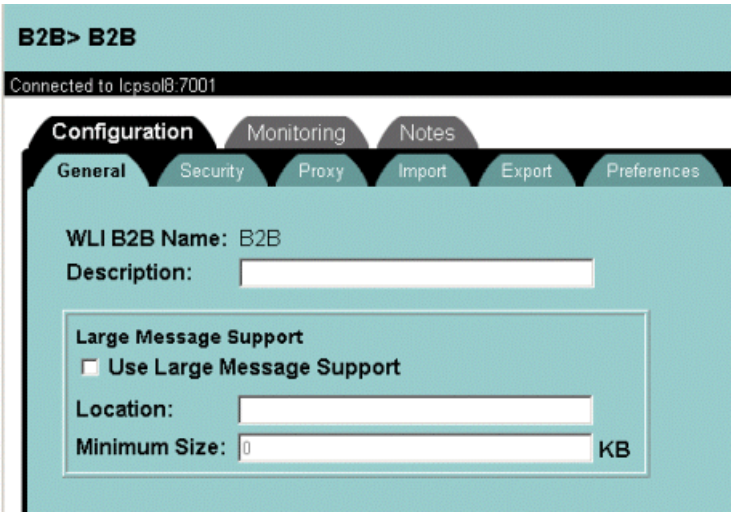
You can tune WebLogic Integration performance by configuring the resource connection pools for J2EE Connector Architecture (J2EE-CA) adapters, which are described in "J2EE Connector Architecture" on page 1-8. For instructions on how to tune resource connection pools for a particular adapter, see the documentation for the adapter.

Configuring Large Message Support for B2B

If the messages exchanged during B2B conversations are too large to fit in memory, enable large message support on the WebLogic Integration B2B Console and restart the server. (Messages larger than 20MB are considered large messages.) Figure 6-1 shows the portion of the B2B Console used for enabling large message support.

Note: For information about configuring EJB transactions for large messages, see "Configuring EJB Transactions" on page 6-9.

Figure 6-1 Enabling Large Message Support in the B2B Console



Configuring EJB Transactions

If your system returns an exception indicating that a transaction timed out while a message was being processed, we recommend that you tune the transaction timeout parameters in the following BPM resources:

- WLI-BPM Server (`wlpi-ejb.jar`)
- WLI-BPM Event Processor EJB (`wlpi-mdb-ejb.jar`)

Note: Transaction timeouts are more likely to occur when large messages, rather than small messages, are being processed.

To tune the transaction timeout parameters, we recommend that you change the `trans-timeout-seconds` attribute in the `wlpi-ejb.jar` and `wlpi-mdb-ejb.jar` files from 90 seconds to 1090 seconds. To access the JAR files:

1. In the Administration Console navigation tree, select Deployments→EJB.
2. Select WLI-BPM Server EJB or WLI-BPM Event Processor EJB.
3. Click Edit EJB Descriptor to display a new window in which you can edit the EJB descriptor.

Monitoring and Tuning the Java Virtual Machine (JVM)

WebLogic Integration Java code is executed on the Java Virtual Machine (JVM). To achieve the optimal performance for a WebLogic Integration deployment, you need to tune the JVM configuration. For example, the JVM heap size determines how often and for how long the JVM collects garbage. For WebLogic Integration, the recommended minimum heap size is 512KB. For more information about configuring the JVM, see "Tuning Java Virtual Machines (JVMs)" in *BEA WebLogic Server Performance and Tuning* at the following URL:

`http://edocs.bea.com/wls/docs70/perform/JVMTuning.html`

For more information about the Sun HotSpot JVM heap organization and garbage collection, go to the following URL:

`http://java.sun.com/docs/hotspot/gc/index.html`

For a complete list of command-line options for the Sun Hotspot JVM, go to the following URL:

`http://java.sun.com/docs/hotspot/VMOptions.html`

Many of the JVM options are set in `setenv.cmd` or `setenv.sh` and `startWeblogic.cmd` or `startweblogic.sh`. Some defaults are set low in order to enable low-end systems. If you have a larger system, you can benefit from tuning the JVM up. The following sections explore commonly used options.

Choosing the JVM

The JDK supplied with WebLogic Server supports different JVM implementations. On Solaris systems, we recommend that you use the server JVM. Use the `-server` argument to specify the use of the server JVM. This argument must be the first one immediately after the Java executable name.

On Windows systems, use the Hotspot JVM (the `-hotspot` option is used by default in WebLogic Integration scripts). If you experience problems using the Hotspot JVM on Windows systems, we recommend adding the following option to your scripts:

```
-xxMaxPermSize=131072K
```

The classic JVM is not recommended because it does not provide a JIT compiler. The server runs more slowly with the classic JVM than with the Hotspot or server JVM.

The Hotspot and server JVM are identical except that they use different run-time compilation algorithms. (The Hotspot JVM is also known as the client JVM.)

Tuning JVM Heap Size

The minimum (initial) and maximum sizes should be identical. For a large WebLogic Integration server, we recommend 512Mb for both values, as shown in the following option settings:

```
-Xms512m –Xmx512m
```

On Solaris systems, there are extra options that apply to very large heaps. In particular, it is possible to bypass virtual memory and use physical memory directly for the heap. This feature is called "Intimate Shared Memory," and information about it can be found at:

http://java.sun.com/docs/hotspot/ism.html

Garbage Collection Control on Hotspot JVM

The heap space in Hotspot is defined in two parts: the *nursery* heap space and the tenured heap space.

All new objects are created in the nursery heap space. They are moved to the tenured heap only after surviving garbage collection from the nursery heap. The tenured heap is not collected as often as the nursery heap, and the collection operation for it is more expensive than collection for the nursery heap.

In general, the nursery heap should be configured to be large enough to store temporary objects. In the case of an application server in general, and for WebLogic Integration in particular, the actual application state is kept in a database. Most memory allocated while a request is being processed is released at the end of the request. It is therefore important to configure the nursery heap to be large enough to prevent objects that are used in a single request to be moved to the tenured heap. Such a configuration also delays the need for collection on the tenured heap, which is much slower than collection on the nursery heap. (For this reason, this approach is sometimes referred to as delayed garbage collection).

Garbage collection in the nursery heap is generational. In generational garbage collection, all new objects are allocated from a nursery heap space. All the objects in the nursery heap space constitute a *young generation* of objects. When the nursery heap space is full, the garbage collector does a partial garbage collection. It reclaims

memory in the nursery space for objects that are no longer accessible, that is, *dead* objects. Objects in the nursery space that are still *live* are moved to an area of memory for older objects. Generational garbage collection can be much faster than full garbage collection because the garbage collector does not have to search all of memory for dead objects.

For more information about garbage collection and JVM performance, see *A Test of JavaTM Virtual Machine Performance*, which is available at the following URL:

http://developer.java.sun.com/developer/technicalArticles/Programming/JVMPerf/

When you have a global heap of 512MB, a reasonable size for the nursery heap is 128MB. The specifications in the following line set the initial nursery heap size to 128 MB, and the maximum size to 128 MB, respectively:

```
-XX:NewSize=128m -XX:MaxNewSize=128m
```

The nursery space is composed of an eden space and two equal-size semispaces. During garbage collection, surviving objects are moved to a semispace. The *survivor space* is the combined size of the two semispaces.

You can use the *SurvivorRatio* parameter to tune the size of the survivor space. The initial recommended value for the survivor ratio is 2. You should monitor your application to determine if you need to change this. Use the following option setting to specify a survivor ratio of 2:

```
-XX:SurvivorRatio=2
```

This parameter sets the ratio between the eden space and *each* semispace to equal 2:1. In other words, the ratio of eden space to survivor space is 1:1; each semispace is one quarter the size of the nursery heap (not one half, because there are two semi spaces of equal size). If survivor spaces are too small, copying collection overflows directly into the old generation. If survivor spaces are too large, they are uselessly empty.

Monitoring JVM Heap Usage

The most efficient way to monitor heap usage and garbage collection is to use verbose garbage collection, selected by specifying the following flag:

```
-verbosegc
```

The output shows up on standard out. In the case of the Hotspot JVM two types of lines show up, indicating collection in the eden (GC) or in the tenured heap (Full GC).

It is also possible to use the Weblogic Server Administration Console to monitor heap utilization at run time. This helps define the heap requirements as well as identifying any memory leaks.

Monitoring and Tuning Run-Time Performance

The following sections describe how to monitor run-time performance in a WebLogic Integration deployment:

- Monitoring and Tuning WebLogic Server Performance
- Monitoring and Tuning BPM Performance
- Monitoring and Tuning B2B Integration Performance
- Monitoring and Tuning Application Integration Performance
- Profiling Applications

Monitoring and Tuning WebLogic Server Performance

Use the WebLogic Server Administration Console to monitor the health and performance of your WebLogic Server domain, including such resources as servers, JDBC connection pools, JCA, HTTP, the JTA subsystem, JNDI, and EJBs. For detailed information, see "Monitoring a WebLogic Domain" in the *Creating and Configuring WebLogic Server Domains* at the following URL:

`http://edocs.bea.com/wls/docs70/admin_domain/monitoring.html`

Do You Have Enough Threads?

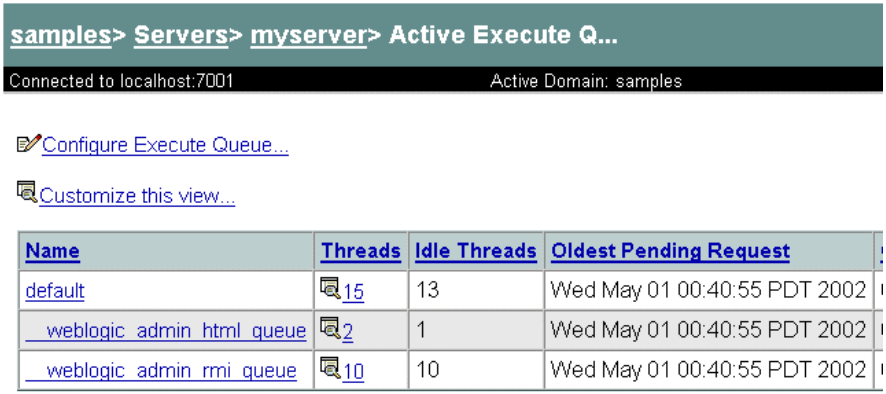
To determine whether your system has enough threads configured:

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

2. Select the Monitoring tab, followed by the General tab.
3. Click Monitor All Active Queues.

   The following figure shows how the WebLogic Server Administration Console displays information about active queues.

Figure 6-2 Active Execute Queues Table

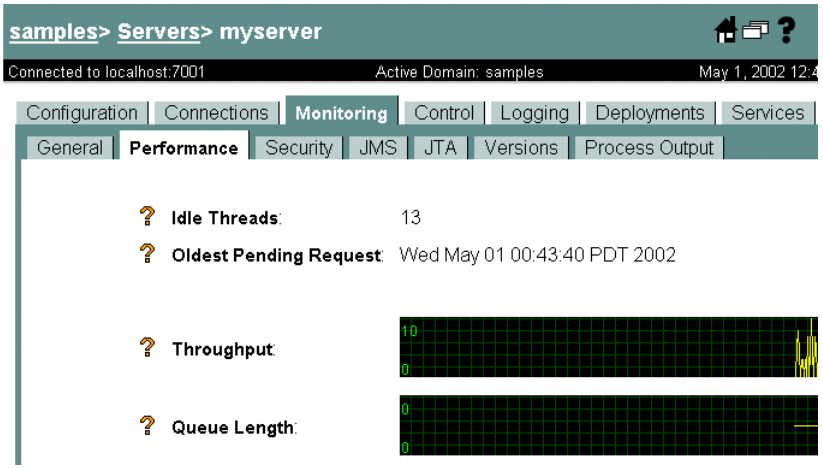


The ThreadPoolSize parameter controls the number of threads. The ThreadPoolSize parameter is set separately for each server:

1. In the WebLogic Server Administration Console navigation tree, select a server name.
2. Select the Monitoring tab, followed by the Performance tab.

   Three graphs are displayed: Throughput, Queue Length, and Memory Usage. The Idle Threads field is displayed above the graphs. The following figure shows how the WebLogic Server Administration Console displays performance information.

Figure 6-3 Server Performance Information



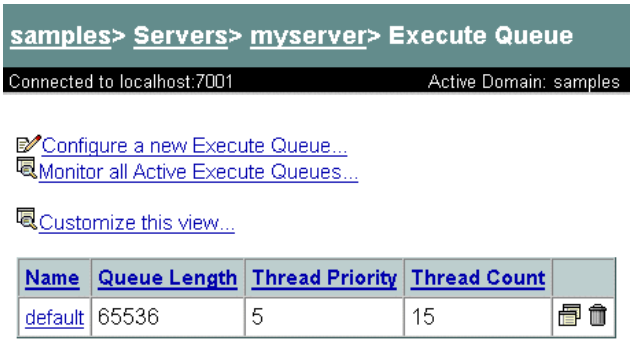
3. If you determine that the number shown in the Idle Threads field is sometimes zero, you need more threads.

To add more threads, you must select the default queue and specify the thread count as follows:

1. In the WebLogic Server Administration Console navigation tree, select a server, then the Monitoring—General tab.
2. Select Monitor all Active Queues.
3. Select Configure Execute Queue.

   The WebLogic Server Administration Console displays execute queue information as shown in the following figure.

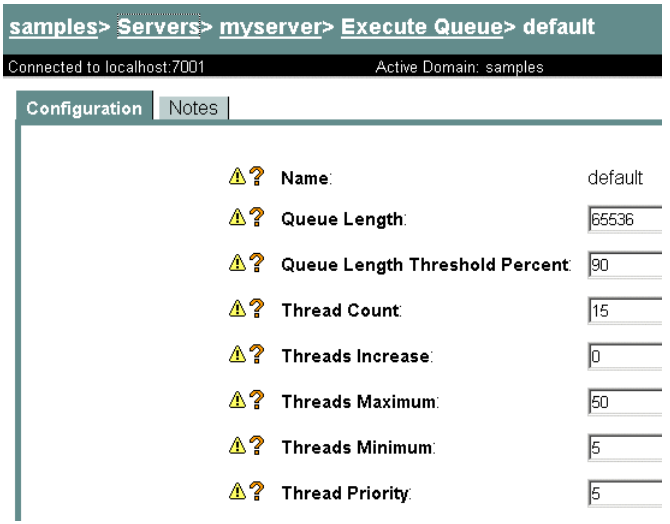
Figure 6-4 Execute Queue Table



You can specify the thread count as follows:

1. In the WebLogic Server Administration Console navigation tree, right-click on a server name to display a drop-down menu.
2. Select View Execute Queues from the drop down menu to display a window, as shown in the following figure. Specify the thread count.

Figure 6-5 Default Execute Queue Configuration



On Solaris systems, you can also determine whether changing the number of threads improves performance by running the `mpstat` command at comparable load levels before and after you change the setting. A decrease in the number of context switches suggests that performance has improved.

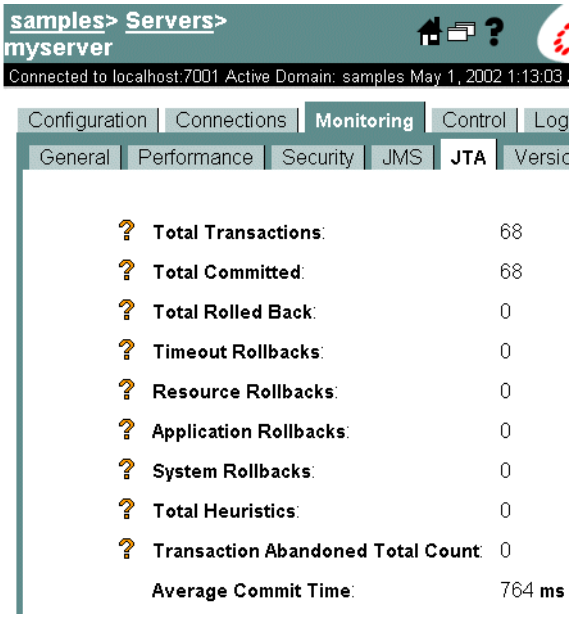
How Many Transactions Are Occurring?

To display the number of transactions of various types, select your server name in the Weblogic Server Administration Console. In the right frame, select the Monitoring tab, then the JTA tab. Select `Monitor all instances`.

Some transactions are associated with the BPM framework; they cannot be changed. Transactions associated with your applications can be changed, however. You can change transaction types or combine transactions by completing the following steps:

1. In the Administration Console navigation tree, select a server.
2. Select the Monitoring tab, then the JTA tab to display the window used to monitor transactions, as shown in the following figure.

Figure 6-6 JTA Monitoring Tab

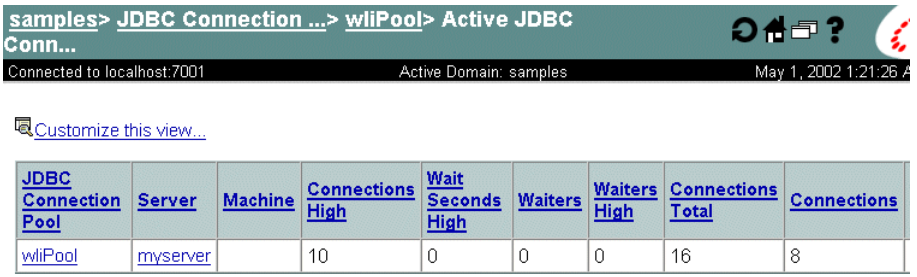


Do You Have Enough JDBC Connections?

JDBC connections are connections to your database, made available so that individual threads do not suffer performance problems caused by getting a new connection every time access to the database is required. You may have multiple pools of JDBC connections. It is important that each pool has enough connections so that no thread has to wait long for a connection. To monitor active JDBC connection pools:

1. In the WebLogic Server Administration Console navigation tree, select Services.
2. Choose JDBC→Connection Pools to display all JDBC connections in the main window.
3. Click the name of a pool, and then click Monitor All Active Pools to display Active JDBC Connections, as shown in the following figure.

Figure 6-7 Active JDBC Connection Pools

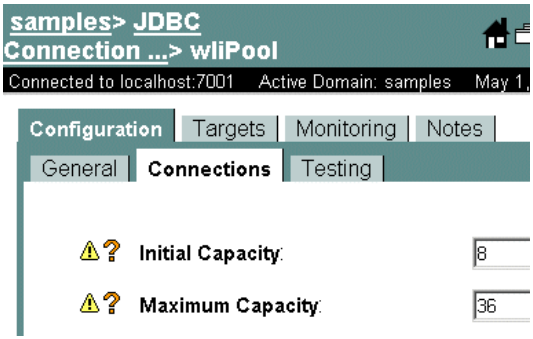


4. Look at the number of Connections; is it close to the total number of connections configured for this pool? Is the High Connections value equal to the total number of connections configured for this pool? Either of these is a sign that more connections may prove useful under similar situations or when load increases slightly.

To modify connection pool configuration:

1. In the Administration Console navigation tree, select Services.
2. Choose JDBC→Connection Pools→wliPool.
3. In the main window, select the Connections tab.

Figure 6-8 Connection Pool Configuration



4. Set the values for the Initial Capacity and Maximum Capacity fields to the same number.

Monitoring and Tuning BPM Performance

Use the WebLogic Integration Studio to monitor various aspects of workflow performance in real time, including the status of workflows and workflow variables. The Studio allows you to delete workflow instances and to view reports on workloads and performance statistics. For more information, see "Monitoring Workflows" in *Using the WebLogic Integration Studio*.

Key BPM performance measurements include:

- *Instantiations*—The number of workflows started within a given time period. Instantiations include operations that are executed *by concurrent clients*: instantiating the workflow, executing the task, and sending an event to the server.
- *Completions*—The number of workflows completed (as indicated by arrival at a Done node) within a given time period. Completions include operations that are executed *from the server side*: instantiating the workflow, executing the task, receiving an event from the client, performing the business operation, and marking the task as done.
- *Guaranteeing message delivery*—Using key BPM features, as necessary, to ensure that messages are sent to target workflows and not lost

One way to obtain statistics for these performance measurements is to extract them from the database instance table using SQL statements. For example, the SQL code in the following listing calculates statistics about the number of instantiations.

Listing 6-1 SQL Code to Determine Workflow Instantiation Statistics

```
select 'INSTANTIATIONS', count(*),
avg((completed-started)*86400),
max((completed-started)*86400),
86400*(max(started)-min(started)) total_duration,
from instance
```

The SQL code in the next listing calculates statistics about the number of completions.

Listing 6-2 SQL Code to Determine Workflow Completion Statistics

```
select 'COMPLETIONS',  count(*),
avg((completed-started)*86400),
max((completed-started)*86400),
86400*(max(completed)-min(started)) total_duration
from instance where completed is not null
```

Do You Have Enough Message-Driven Beans?

To display information about message-driven beans:

1. In the WebLogic Server Administration Console navigation tree, select a server name.
2. Select the Monitoring tab, followed by the JMS tab.
3. Click Monitor all Active JMS Servers.
4. Click on the number of Active JMS Destinations. The Active JMS Destinations are displayed, as shown in the following figure.

Figure 6-9 Event Queue Monitoring

Monitor all Durable Subscribers...

Customize this view...

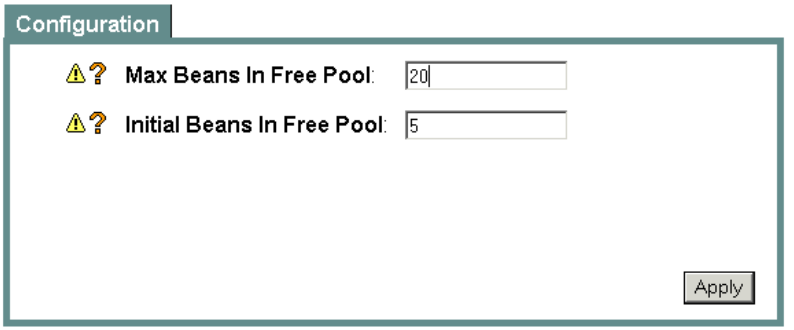
Name	Consumers	Consumers High	Consumers Total	Messages	Messag Pending
timerQueue	13	13	13	0	0
eventQueue	10	10	10	0	0
wlpiValidatingEvent	10	10	10	0	0

5. Look at the queue length (messages pending) for WLI_BPM_Event. If the number is often more than just a few, it is safe to conclude that the amount of queueing that is occurring is greater than an amount that is consistent with good performance. In this case, adding more message-driven beans helps performance.

To change the number of message-driven beans:

1. In the Administration Console navigation tree, select Deployments—EJB.
2. Select WLI-BPM Event Processor.
3. Click Edit EJB Descriptor to display a new window, in which you can edit the EJB descriptor.
4. In the left navigation window, select WebLogic EJB Jar—WebLogic Enterprise Bean—EventListener—Message Driven Destination—Pool to display the configuration window shown in the following figure.

Figure 6-10 Configuring MDBs



5. Edit the value of the Max Beans in Free Pool parameter.
6. Reboot Weblogic Server for this change to take effect.

How Many of Each Type of Bean Does My System Have?

Use the WebLogic Server Administration Console to display information bean types and quantities:

1. In the Administration Console navigation tree, select Deployments—EJB.
2. Select the name of a particular EJB. For example, select WLI-BPM Server.
3. In the main window, select the Monitoring tab and the type of bean to be displayed. For example, to display information about stateful session beans, click Monitor all Stateful EJBRuntimes.
4. To modify the display of information, click Customize this view. You can add or delete columns and change the sort order.

   The following columns are of particular interest:

   - Number of beans in use
   - Number of beans in cache

   The following JAR files are of particular interest:

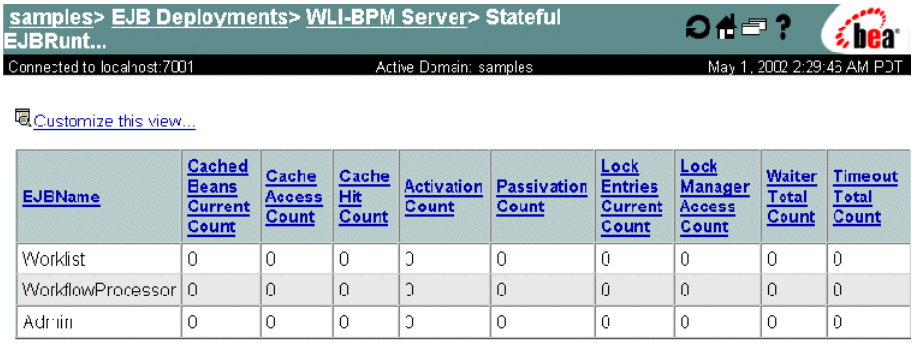
   - `wlpi-ejb.jar`
   - `wlpi-mdb-ejb.jar`
   - `wlai-server-ejb.jar`

- `wlai-eventprocessor-ejb.jar`
- `wlai-asyncprocessor-ejb.jar`
- The JAR files for your application-specific EJBs

For more information about these JAR files, see "Configuring EJB Pool and Cache Sizes" on page 6-3. The following figures (Figure 6-11, Figure 6-12, and Figure 6-13) show portions of the windows in which information for stateful, entity, and message-driven beans is displayed.

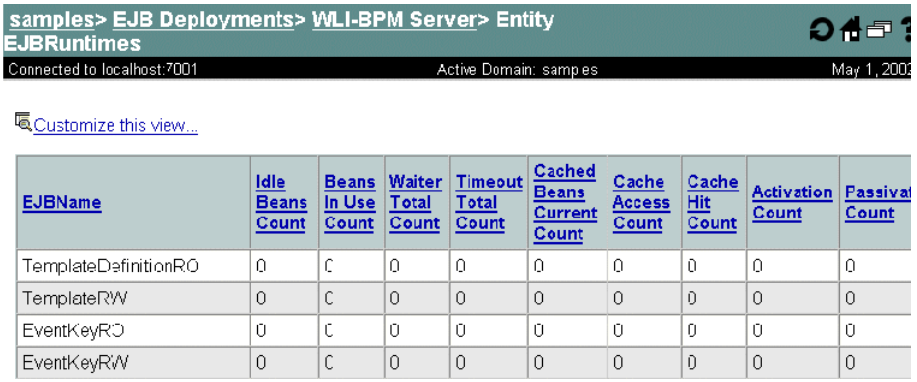
The following figure displays Stateful EJBRuntimes for the WLI-BPM Server EJB (`wlpi-ejb.jar`).

Figure 6-11 Stateful Bean Information



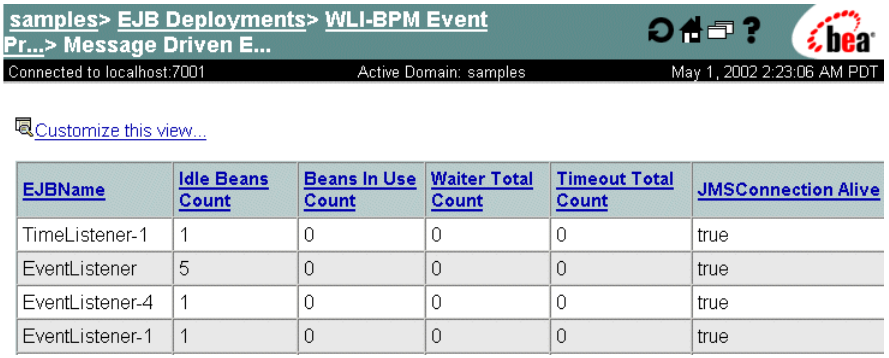
The following figure displays Entity EJBRuntimes for the WLI-BPM Server EJB (`wlpi-ejb.jar`).

Figure 6-12 Entity Bean Information



The following figure displays Message Driven EJBRuntimes for the WLI-BPM Event Processor EJB (`wlpi-mdb-ejb.jar`).

Figure 6-13 MDB Information



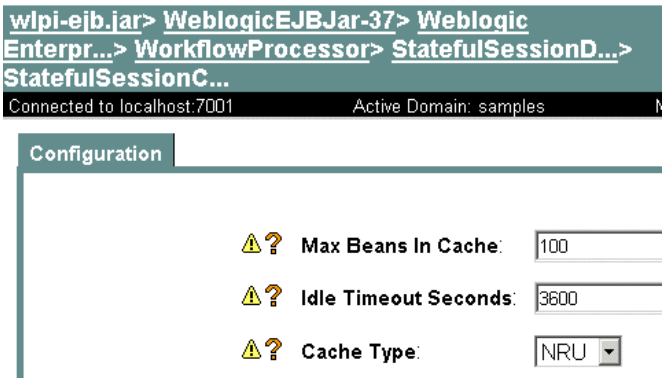
If a system message concerning *cache full* is displayed, increase the Max Beans in Cache parameter for the corresponding EJB by editing the EJB descriptor.

If many entity beans are not passivated until the cache is full, you may want to decrease the Idle Timeout Seconds parameter for the entity bean:

1. In the Administration Console navigation tree, select Deployments—EJB.
2. Select the name of a particular EJB. For example, select WLI-BPM Server.

3. Click Edit EJB Descriptor to display a new window, in which you can edit the EJB descriptor.
4. In the left navigation window, select WebLogic EJB Jar→WebLogic Enterprise Bean→WorkflowProcessor→Stateful Session Descriptor→Stateful Session Cache to display the configuration window shown in the following figure.
5. Edit the Idle Timeout Seconds parameter.

Figure 6-14 Idle Timeout Configuration



Guaranteeing Message Delivery

Depending on the design requirements for your business processes, you may want to take advantage of two features that can guarantee the delivery of a message to a workflow. The features summarized in this section apply specifically to messages sent from any JMS client to a workflow, which includes workflow-to-workflow, and not to business messages sent between trading partners (trading partner business messages use addressed messaging by default).

The following features guarantee the delivery of messages:

- Addressed messaging, which allows you to specify the ID of a specific workflow instance to which the message is to be sent. Using addressed messages can also guarantee that a response message is delivered to a particular workflow instance that has begun a conversation with the current workflow (either by calling it via the Start Workflow action, or by triggering a Start or Event node contained

within it via a previously sent XML message) — even if the receiving Event node in the instantiated workflow has not yet been activated in the flow.

- Setting the transaction mode in the Post XML Event action so that the message is sent only if the transaction completes successfully and a commit is issued. By making sure that the message is not sent until the workflow enters its quiescent state, this setting ensures that the resources required to guarantee message delivery are allocated and committed.

Using these two features together ensures that the delivery of a message to a workflow is guaranteed. For information about using these features, see the following:

- "Guaranteeing Message Delivery" in "Establishing JMS Connections" in *Programming BPM Client Applications*.
- "Posting an XML Message to a JMS Topic or Queue" in "Defining Actions" in *Using the WebLogic Integration Studio*.

For an example of using addressed messaging, see "Business Process and Workflow Modeling" in "Understanding the Sample" in *Learning to Use BEA WebLogic Integration*.

For information about delivery of business messages between trading partners, see *Creating Workflows for B2B Integration*.

Monitoring and Tuning B2B Integration Performance

To monitor the performance of B2B integration functionality, consider the following tips:

- Use the WebLogic Integration B2B Console to monitor and control aspects of B2B integration functionality, including trading partner sessions, delivery channels, conversations, and collaboration agreements.
- To monitor run-time performance, inspect `access.log`, the file used for tracking the arrival of HTTP requests in the system. This file enables system administrators to validate the state of the network/TCP interface. The time stamps give a good indication of the rate of arrival of requests.
- To detect a bottleneck in the flow of an XOCP message, use the `getHopTimestamps()` method of the `QualityOfService` class on the consumer trading partner side. This method returns timestamps at all the hops of

the message. To interpret the data accurately, ensure that the clocks in all the machines are synchronized.

Key performance measurements for B2B integration include:

- *Throughput*—The number of messages processed by the hub (sent and received) during a given time period.
- *Trip Time*—Amount of time required for a request to travel from one spoke to another through the hub.

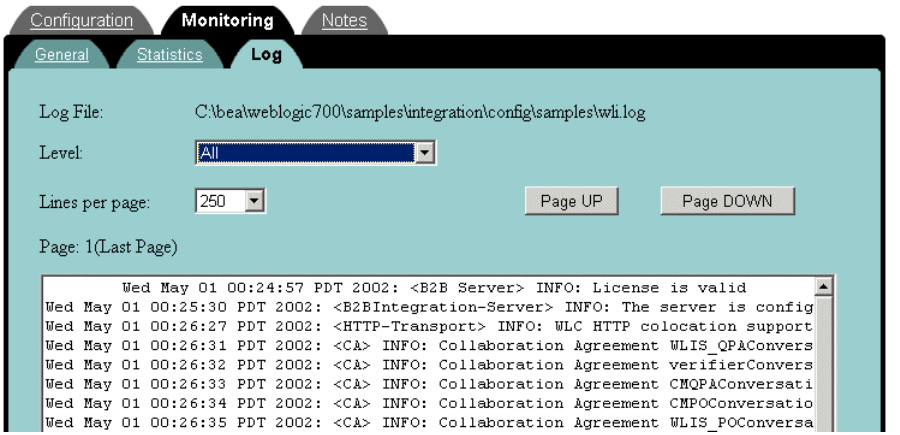
For more information, see "Monitoring B2B Integration" in *Administering B2B Integration*.

Monitoring B2B Activity

Use the WebLogic Integration B2B Console to determine the level of B2B activity:

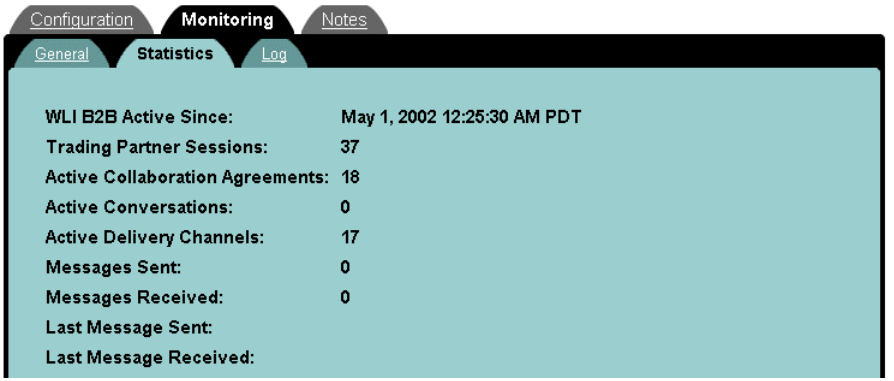
1. In the B2B Console navigation tree, select the parent node: B2B.
2. In the main window, select the Monitoring tab, followed by the Log tab to display the `wli.log` file, as shown in the following figure.

Figure 6-15 Monitoring B2B Logs



3. In the main window, select the Monitoring tab, followed by the Statistics tab to display B2B statistics, as shown in the following figure.

Figure 6-16 Monitoring B2B Statistics



Monitoring and Tuning Application Integration Performance

This section provides information about monitoring and tuning application integration. It contains the following topics:

- Monitoring and Tuning Application View Connections
- Monitoring and Tuning EJB Pools for Application Integration

Monitoring and Tuning Application View Connections

You can check if you have sufficient connections available for your application view, using the Weblogic Server Administration Console:

1. In the Administration Console navigation tree, select Deployments→Connectors.
2. Select the connection factory deployed for your application view, which is named using the following format:

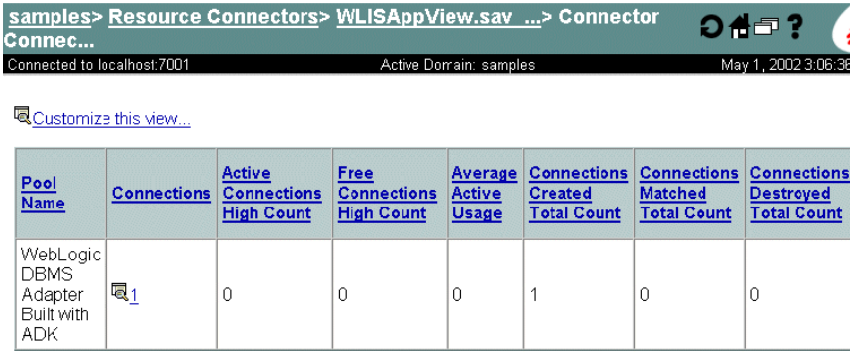
   *ApplicationViewName*_connectionFactory.

3. Select the Monitoring tab, and click Monitor all Connector Connection Pool Runtimes...

The connections to the EIS defined in your application view are displayed, as shown in the following figure.

These connections are made available so that individual threads do not suffer performance problems caused by getting a new connection every time access to the EIS is required. It is important that each has enough connections so that no thread has to wait long for a connection.

Figure 6-17 Monitoring Application View Connection



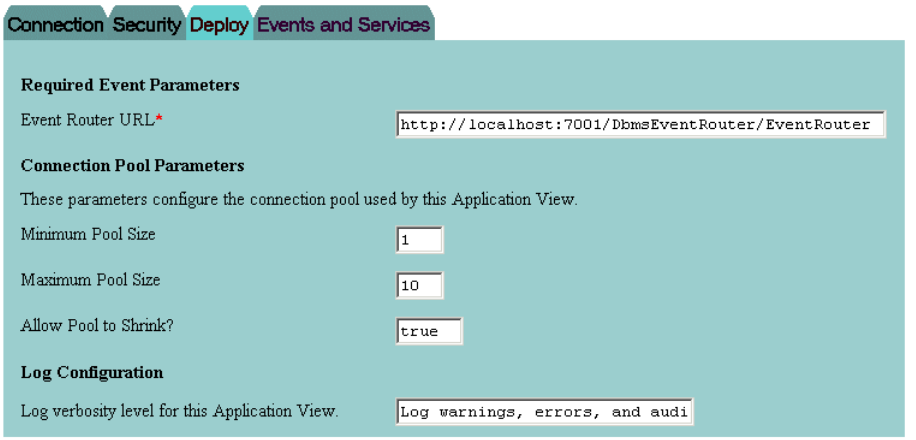
4. Look at the number of connections:
   - Is it close to the total number of connections configured for this pool?
   - Is the Active Connections High Count value equal to the total number of connections configured for this pool?

   If either case is true, increasing the number of connections will be useful in similar situations or when load increases slightly.

To view or modify your maximum connections for your application view:

1. Start the Application View Console
2. Select your application view, and then select the Deploy tab. The following figure shows the Application View Console tab used to monitor the maximum pool size.

Figure 6-18 Monitoring Maximum Pool Size



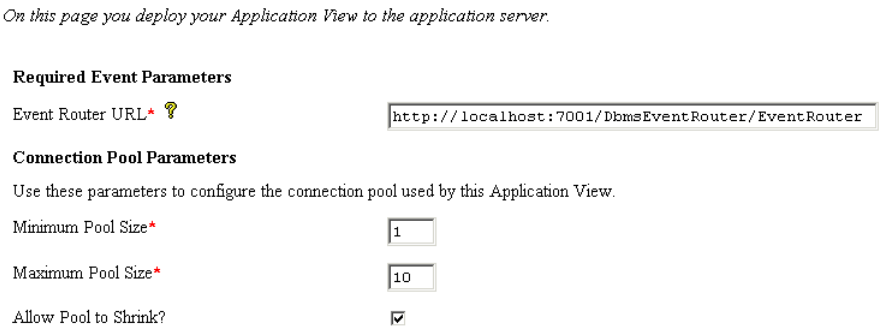
The Maximum Pool Size value shows the maximum number of connections.

To modify the maximum pool size value, complete the following steps:

1. Click Undeploy if the Application View is currently deployed.
2. When the Application View is undeployed, click Edit.

   A window in which you can edit events or services is displayed.
3. Click Continue to display the window shown in the following figure, in which you can edit the maximum pool size.

Figure 6-19 Modifying Maximum Pool size



4. Edit the Maximum Pool Size value (the maximum number of connections).
5. Click Deploy to redeploy the Application View with the new Maximum Pool Size value.

Monitoring and Tuning EJB Pools for Application Integration

If you want to tune your application integration performance, consider tuning the following EJB pools:

- Asynchronous service processor message-driven bean pools (`wlai-asyncprocessor-ejb.jar`)
- Event processor message-driven bean pools (`wlai-eventprocessor-ejb.jar`)
- Application View stateful and stateless session EJB pools (`wlai-server-ejb.jar`)

For information about monitoring and tuning EJB pools, see "Do You Have Enough Message-Driven Beans?" on page 6-20, and "How Many of Each Type of Bean Does My System Have?" on page 6-22.

Profiling Applications

You can profile applications at run time using a Java profiler tool (such as Jprobe or OptimizeIt). Use these tools to identify performance bottlenecks and thread contentions in the system. Remember to profile run-time performance rather than boot-time performance.

Tuning Hardware, Operating System, and Network Resources

The following sections describe factors that you need to consider when you are tuning hardware, the operating system, and the network:

- Tuning Hardware
- Tuning the Operating System
- Tuning Network Performance

For detailed information, see "Tuning Hardware, Operating System, and Network Performance" in *BEA WebLogic Server Performance and Tuning* at the following URL:

`http://edocs.bea.com/wls/docs70/perform/HWTuning.html`

Performance Bottlenecks

To optimize WebLogic Integration performance in a deployment, you need to understand how the following hardware resources interact with each other. Performance bottlenecks result from poor tuning of these hardware resources.

Table 6-3 Performance Bottlenecks

Hardware Resource	Bottlenecks
CPU	Insufficient throughput, resulting in excessive paging and swapping.
Memory	Insufficient system memory, resulting in excessive paging and swapping.
Network resources	Insufficient bandwidth to handle high volumes of network traffic. A high frequency of network collisions.
Disk I/O and controllers	Insufficient capacity and throughput to handle the volume and size of I/O requests.

Tuning Hardware

To optimize WebLogic Integration performance in a deployment, consider the following hardware factors:

- Number of machines (as well as the number of CPUs per machine) required to run WebLogic Integration during average and peak loads at acceptable performance levels
- Right kind of storage, configuration, and acceptable size. To enhance RDBMS performance, use faster disks.
- Amount of main memory required to handle average and peak loads at acceptable performance levels

Tuning the Operating System

To optimize WebLogic Integration performance in a deployment, consider the following operating system factors:

- Configurable file descriptor limits
- Memory allocation for user processes
- Configurable TCP tuning parameters
- Configurable settings for the threading model
- Use of monitoring tools such as vmstat, mpstat, netstat, iostat, and so on

Configurable TCP Tuning Parameters on Windows NT/2000

For a Windows NT or Windows 2000 server, we recommend setting the `TcpTimedWaitDelay` parameter to 60 seconds instead of the default 240 seconds. The parameter is in the Windows registry and can be set or modified by using the `regedit` utility (`regedit.exe`). The entry is located as follows:

```
HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\Tcpip\Parameters
```

The entry is not present by default.

TcpTimedWaitDelay determines the time that must elapse before TCP can release a closed connection and reuse its resources. This period between closure and release is known as the TIME_WAIT state or 2MSL state. During this time, the connection can be reopened at much less cost to the client and server than the cost of establishing a new connection.

RFC 793 requires that TCP maintain a closed connection for an interval at least equal to twice the maximum segment lifetime (2MSL) of the network. When a connection is released, its socket pair and TCP control block (TCB) can be used to support another connection. By default, the MSL is defined to be 120 seconds, and the value of this entry is equal to two MSLs, or 4 minutes. However, you can use this registry entry to customize this interval.

Reducing the value of this entry allows TCP to release closed connections faster, providing more resources for new connections. However, if the value is too low, TCP might release connection resources before the connection is complete, requiring the server to use additional resources to reestablish the connection.

Note: Normally, TCP does not release closed connections until the value of this entry expires. However, TCP can release connections before this value expires if it is running out of TCP control blocks (TCBs). The number of TCBs the system creates is specified by the value of MaxFreeTcbs.

System Monitoring on Windows NT/2000

Use the performance monitor (perfmon.exe) for monitoring all system resources or the task manager for monitoring CPU, memory, and threads.

Swap Space Configuration for Solaris

Insufficient swap space can show up as an out-of-memory error, such as an overly small heap or thread limit.

Network Tuning for Solaris

For network tuning information for Solaris systems, see the WebLogic Server platform information page at the following location:

http://e-docs.bea.com/wls/platforms/sun/index.html

System Monitoring for Solaris

The following table lists the commands suggested for use in monitoring Solaris systems.

To monitor . . .	Use . . .
Memory utilization	`vmstat`
CPU utilization	`mpstat 5`. (In addition to CPU utilization, this command also displays the context switches on a per-processor basis. For aggregate CPU utilization, use the `sar` command.
Disk I/O	`iostat`
Network I/O	`netstat -sP tcp`. This command monitors the various TCP parameters.

Tuning Network Performance

To optimize WebLogic Integration performance in a deployment, consider the following requirements for a high-performance network:

- Sufficient available network bandwidth for WebLogic Integration and its connections to other tiers in your architecture (such as client and database connections)
- Sufficient throughput speed on the LAN/WAN
- Configurable operating system settings that allow you to optimize network performance
- Sufficient capacity to handle peak loads

Tuning Databases

To optimize WebLogic Integration performance in a deployment, you need to maximize the use of underlying resources. WebLogic Integration relies extensively on database resources for handling run-time operations and ensuring that application data is durable. The following sections describe how to tune databases in a WebLogic Integration deployment:

- General Database Tuning Suggestions
- Tuning Oracle Databases
- Tuning Microsoft SQL Server Databases
- Tuning Sybase Databases

These sections provide a checklist of issues to consider when you are working to optimize your WebLogic Integration performance. For detailed instructions about specific database products, consult the appropriate product documentation.

General Database Tuning Suggestions

The following sections explain how you can optimize database performance by adjusting the settings for various parameters and features of your deployment:

- Opened Cursors
- Disk I/O Optimization
- Database Sizing and Organization of Table Spaces
- Checkpointing
- Database Compatibility
- Database Monitoring

Opened Cursors

While using multiple cursors for an operation can increase concurrency in most situations (for example, one opened cursor can perform updates while another opened cursor performs inserts), there is a limit to the maximum number of cursors that can be handled by a database server. This maximum pool is shared across all sessions and connections of the database server. Keeping too many cursors opened within a single connection can starve other connections, thereby slowing database performance and reducing system scalability. A good estimate can be derived from the maximum number of opened cursors that the database server can handle and the average number of simultaneous users. Another strategy is to minimize the length of time that each cursor is kept open.

Disk I/O Optimization

Disk I/O optimization is a key database tuning parameter that is related directly to throughput and scalability. Access to even the fastest disk is orders of magnitude slower than memory access. Whenever possible, optimize the number of disk accesses. In general, selecting a larger block / buffer size for I/O reduces the number of disk accesses and might substantially increase throughput in a heavily loaded production environment.

For recommended settings, see the appropriate database-specific sections about tuning databases, which are provided later in this document.

Database Sizing and Organization of Table Spaces

Distribute the database workload across multiple disks to avoid or reduce disk overloading. To optimize database performance:

- Put frequently accessed tables and indexes on different disks. The mechanism to achieve this differs from database to database. Consult your local database administration guide on organization of database storage structures.

  For example, each workflow instance and its children create a row in the `WORKFLOWINSTANCE` table. These tables need to be optimized for insert and update operations. Delete operations on this table are performed in batches through the WebLogic Integration Studio. For a batch delete operation used to remove workflow instances, be sure to configure a rollback segment with a sufficient size so that it can handle a delete operation.

- Put the redo logs, archive logs, and database tables on separate disks.
- Some databases allow users to choose between raw disk I/O and regular file system I/O. In general, raw disk I/O has better write performance while file system I/O has better read performance due to OS-level caching. Thus raw disk I/O is a good candidate for OLTP applications while file system I/O should be used for decision support applications. When using raw I/O, you should increase the database buffer cache size to compensate for the lack of OS-level caching.

Checkpointing

Checkpoint is a mechanism that periodically flushes all dirty cache data to disk. This increases the I/O activity and system resource usage for the duration of the checkpoint. While frequent checkpointing can increase the consistency of on-disk data, it can also slow database performance. While most database systems have the notion of checkpoint, not all database systems provide user-level controls. Oracle, for example, allows administrators to set the frequency of checkpoints while users have no control over SQLServer 7.X checkpoints. For recommended settings, see the product documentation for the database you are using.

Database Compatibility

Use only the recommended versions of clients and servers. For a list of supported databases, see the software requirements in the *BEA WebLogic Integration Release Notes* for the release of WebLogic Integration that you are using.

Database Monitoring

Monitor the following aspects of database use:

- *Disk space*—Monitor the system to ensure that the database is not running out of space. The key tables, such as `WORKFLOWINSTANCE`, should be monitored to make sure enough space is allocated. Schedule regular table reorganization for space defragmentation (after first monitoring tables for fragmentation) and reclaim the disk space.
- *Performance*—Use the profiling or monitoring tools that accompany your database to identify bottlenecks and to obtain recommendations for performance tuning.

Tuning Oracle Databases

This section describes performance tuning for Oracle 8.1.7.

V$Tables

Oracle 8.1.7 offers a series of dynamic performance views, often called V$tables, that allows users to monitor system statistics using SQL queries. Users need to be logged in to the database as `SYS` or `SYSTEM`, or they must have administrator privileges to access these dynamic views. Many of these dynamic views are referenced in the following sections. For details about these dynamic views, see your Oracle administrator's guide and tuning guide for details.

Initialization Parameters

The initialization parameter file (`init.ora`) contains the system initialization parameters and values for the Oracle server.

On Windows NT/2000, the pathname for the file is as follows:

`d:\oracle\admin\sid\pfile\init.ora`

In this pathname, *`d:\oracle`* is the installation directory and *`sid`* is the instance ID of the database (for example, `d:\Oracle\admin\hsundb\pfile\init.ora`).

The contents of this file are organized as attribute-value pairs, such as `PROCESSES = 100`.

You should always make a backup before modifying the file. You must bounce (shut down and restart) the server to reflect any modifications.

Modifications made to this file can and should be verified after bouncing the server. This validation can be done through an SQL statement or an SQL*Plus command. The parameters and their values are stored in a dynamic performance view, `V$PARAMETER`.

The following query validates changes made to the `PROCESSES` parameter. Note that the attribute name is lower case:

```
SELECT name, value FROM v$parameter WHERE name = 'processes'
```

Another method is to use the `SHOW PARAMETERS` *`parameter_name`* command in an SQL*Plus shell. For example, the following command:

```
SHOW PARAMETERS "parameter"
```

is roughly equivalent to the following query:

```
SELECT name, value FROM v$parameter WHERE name LIKE '%parameter%';
```

Ensure that you have a full understanding of the parameter before modifying its value. For detailed information about specific parameters, see your Oracle documentation.

Shared Pool Size

The share pool in an important part of the Oracle server system global area (SGA). The SGA is a group of shared memory structures that contain data and control information for one Oracle database instance. If multiple users are concurrently connected to the same instance, the data in the instance's SGA is shared among the users.

The shared pool portion of the SGA caches data for two major areas: the library cache and the dictionary cache. The library cache is used to store SQL-related information and control structures (for example, parsed SQL statement, locks). The dictionary cache is used to store operational metadata needed for SQL processing.

For most applications, the shared pool size is critical to Oracle performance. If the shared pool is too small, the server must dedicate resources to managing the limited amount of available space. This consumes CPU resources and causes contention because Oracle imposes restrictions on the parallel management of the various caches. The more you use triggers and stored procedures, the larger the shared pool must be.

The `SHARED_POOL_SIZE` initialization parameter specifies the size of the shared pool in bytes. We recommend a value that is no less than 9MB in a production system. It is not uncommon for systems to require up to 75MB for the shared pool. The following query monitors the amount of free memory in the share pool:

```
SELECT * FROM v$sgastat
WHERE name = 'free memory' AND pool = 'shared pool';
```

If there is always free memory available within the shared pool, then increasing the size of the pool offers little or no benefit. Also, just because the shared pool is full does not necessarily mean there is a problem. There are no entries in the shared pool that cannot be paged out once they enter the pool. Application and deployment needs may differ, thus this value needs to be tuned on the basis of specific deployments and applications.

Maximum Opened Cursors

To prevent any single connection taking all the resources in the Oracle server, the `OPEN_CURSORS` initialization parameter allows administrators to limit the maximum number of opened cursors for each connection. Unfortunately, the default value for this parameter is too small for systems such as WebLogic Server and WebLogic Integration. A reasonable number falls in the range of 175 to 255. Cursor information can be monitored using the following query:

```
SELECT name, value FROM v$sysstat
WHERE name LIKE 'opened cursor%';
```

Maximum Number of Processes

On most operating systems, each connection to the Oracle server spawns a shadow process to service the connection. Thus, the maximum number of processes allowed for the Oracle server must account for the number of simultaneous users, as well as the number of background processes used by the Oracle server. The default number is usually not big enough for a system that needs to support a large number of concurrent operations. A reasonable number falls in the range of 200 to 255. For platform-specific issues. see your Oracle administrator's guide. The current setting of this parameter can be obtained with the following query:

```
SELECT name, value FROM v$parameter WHERE name = 'processes';
```

Database Block Size

A block is Oracle's basic unit for storing data and the smallest unit of I/O. One data block corresponds to a specific number of bytes of physical database space on disk. This concept of a block is specific to Oracle RDBMS and should not be confused with the block size of the underlying operating system. Note that since the block size affects physical storage, this value can be set only during the creation of the database; it cannot be changed once the database has been created.

Given the nature of WebLogic Integration repository tables and access patterns, it is recommended that the database used for WebLogic Integration is created with a block size of 8K. The current setting of this parameter can be obtained with the following query:

```
SELECT name, value FROM v$parameter WHERE name = 'db_block_size';
```

The following table shows the advantages and disadvantages of commonly used block sizes.

Block Size	Advantages	Disadvantages
2K-4K (small)	Reduces block contention when multiple transactions act upon the same block. Good for small rows, or lots of random access.	Has relatively large I/O overhead. You may end up storing only a small number of rows in each block, depending on the size of the row.
8K (medium)	If rows are medium size, then you can bring a number of rows into the buffer cache with a single I/O. With a small block size, you may bring in only a single row.	Space in the Oracle buffer cache is wasted if you are doing random access to small rows and have a large block size. For example, with an 8KB block size and 50-byte row size, you are wasting 7,950 bytes in the buffer cache when doing random access.
16K-32K (large)	There is relatively less overhead; thus, there is more room to store useful data. Good for sequential access or very large rows.	Large block size is not good for index blocks used in an OLTP type environment, because they increase block contention on the index leaf blocks.

Tuning Options for System Administrators

This section contains tuning procedures that should be performed only by system administrators or users who are intimately familiar with the affected system.

Warning: Not all the tuning options described in this section have a positive effect on performance; you may need to derive values for parameters empirically.

SNP Processes

By default, the Oracle server creates several background processes to perform scheduled tasks. These tasks can be scheduled only through the use of the Job Queues functionality or Advanced Replication (check your Oracle documentation for details). If you are not using these Oracle features, however, these background processes waste resources. To turn off these processes until they are actually needed, modify the `init.ora` file.

The safest way to do this is by commenting out the following section in your `init.ora` file:

```
# The following parameters are needed for the Advanced Replication
#Option

#job_queue_processes = 4
#job_queue_interval = 10
```

Sort Area Size

Increasing the sort area increases the performance of large sorts as this allows the sort to be performed in memory during query processing. This can be important, as there is only one sort area for each connection at any point in time. The default value of this `init.ora` parameter is usually the size of 6-8 data blocks. This value is usually sufficient for OLTP operations but should be increased for decision support operation, large bulk operations, or large index-related operations (for example, recreating an index). When performing these types of operations, you should tune the following `init.ora` parameters (which are currently set for 8K data blocks):

```
sort_area_size = 65536
sort_area_retained_size = 65536
```

Physical Storage Parameters for Tables

Database tables grow and shrink in size due to inserts, updates, and deletes. Growing a table incurs additional I/O that slows database operations. Thus, the physical storage parameters of each table should be set according to its expected access and usage pattern. This also means that the parameters are largely determined by the applications using the tables. In general, the default values used by Oracle work fairly well, but there are many instances where tuning these parameters can produce dramatic performance improvements. This work should be performed by a professional DBA with a deep understanding of the Oracle RDBMS. The following sections highlight some storage parameters that are common to schema objects, but are especially important to the `CREATE TABLE` command. It is not in the scope of this guide to recommend specific values for these parameters. (For details, see your Oracle documentation or DBA). Selected parameters are described and queries are provided to help you check for potential problems.

- `INITRANS` and `MAXTRANS`

  When a transaction modifies a block, it must first mark a flag in the header of the block. The marker is released when the transaction commits. Each marker

takes space in the block, thus more transaction markers mean less space for data. Without a marker, the transaction is not allowed to modify the block and must wait. Oracle allows users to control the number of markers per block on a per-table basis. (Some tables provide users with an even finer level of control, but a description of such control is beyond the scope of this document.) The INITRANS parameter allows users to specify the initial number of markers allocated in each block (the minimum value is 1). Additional markers are allocated up to the number specified by MAXTRANS. Transactions are blocked when no free markers are available. As transactions become blocked, the possibility of deadlocks increases (that is, transactions that are not allowed to complete and hold on to resource locks). The default MAXTRANS value is 255, but it should be checked with the following query to ensure that the parameters have a reasonable value for tables involved in OLTP:

```
SELECT owner, table_name, ini_trans, max_trans, FROM all_tables;
```

These settings are important if your application involves many concurrent workflows because, during its lifecycle, each workflow executes a series of transactions against the WORKFLOWINSTANCE table.

- MINEXTENTS and MAXEXTENTS

  These parameters control the size of tables as they grow and shrink. An extent is composed of one or more data blocks (see "Database Block Size"). These parameters control the number of extents that are allocated to a table during creation (the size of a table cannot shrink below the value specified by MINEXTENTS) and the maximum number of extents that can be allocated to a table. Generally users should create tables using the following settings:

  ```
  CREATE TABLE foo (col1 number, col2 date)
  STORAGE (MINEXTENTS 1 MAXEXTENTS UNLIMITED);
  ```

  The following query is used to check the values of these parameters:

  ```
  SELECT owner, table_name, min_extents, max_extents
  FROM all_tables;
  ```

  Note that when the UNLIMITED option is specified for MAXEXTENTS, the value returned by the query will be a large integer (for example, 2147483645).

Swapping of Redo Logs

To support recovery, all operations performed against the Oracle RDBMS are recorded in redo logs (unless you explicitly disable logging for certain operations). Over time, the amount of information in the log increases and eventually starts to affect the performance of each operation. Immediately after a successful database backup, the information in the redo logs is no longer necessary as recovery can be achieved with the backup. Thus, it is a good practice to start a new redo log after each backup to clean up the information that is no longer needed and potentially restore system performance. This operation can be done through the following SQL command:

```
ALTER SYSTEM SWITCH LOGFILE
```

For details about redo logging, managing redo logs and log groups, and best practices for RDBMS backup, see your Oracle documentation.

Table Reorganizations

As SQL operations (both OLTP and bulk) cause tables to grow and shrink, the storage space for the table can become fragmented. This can lead to performance degradations and requires user intervention to reclaim space gaps and compact table data. This operation is often referred to as a table reorganization. Oracle 8.1.7 does not have a built-in facility to support this operation, thus the user must perform the steps manually. Following good practices, this operation should be done soon after a database backup. The following steps show how to reorganize a table called `foo`:

1. Make a copy of the table using the following SQL statement:

   ```
   CREATE TABLE foo_bkup AS SELECT * FROM FOO;
   ```

   The act of copying the data will compact the data and since this is a new table, there is no space to reclaim.

2. Delete the old table using the following SQL statement:

   ```
   DELETE TABLE foo;
   ```

3. Rename the new table with the name of the old table using the following SQL statement:

   ```
   RENAME foo_bkup TO foo
   ```

Note that each step in the process involves DDL statements (such as `CREATE TABLE`, `DROP TABLE`, and so on). DDL statements are not transactional in Oracle. More specifically, each DDL statement executes in a self-contained transaction. Thus the `ROLLBACK` command is ineffective during a table reorganization.

Tuning Microsoft SQL Server Databases

The following table describes performance tuning parameters that are specific to Microsoft SQL Server databases. For more information about these parameters, see your Microsoft SQL Server documentation.

Table 6-4 Performance Tuning Parameters for Microsoft SQL Server Databases

Parameter	Recommendation
Tempdb	Store tempdb on a fast I/O device.
Recovery interval	Increase the recovery interval if `perfmon` shows an increase in I/O.
I/O block size	Use an I/O block size larger than 2Kb.

Tuning Sybase Databases

The following table describes performance tuning parameters that are specific to Sybase databases. For more information about these parameters, see your Sybase documentation.

Table 6-5 Performance Tuning Parameters for Sybase Databases

Parameter	Recommendation
Recovery interval	Lower recovery interval setting results in more frequent checkpoint operations, resulting in more I/O operations.
I/O block size	Use an I/O block size larger than 2Kb.

Table 6-5 Performance Tuning Parameters for Sybase Databases (Continued)

Parameter	Recommendation
Maximum online engines	Controls the number of engines in a symmetric multiprocessor (SMP) environment. Sybase recommends configuring this setting to the number of CPUs minus 1.

A Deploying WebLogic Integration Client Applications

WebLogic Integration provides application programming interfaces (APIs) that developers can use to create client applications. Client applications in which WebLogic Integration APIs are used can import WebLogic Integration classes, look up Enterprise Java Beans (EJBs), and so on.

This section provides information to help you deploy your WebLogic Integration client applications successfully.

JAR Files

Specify the following JAR files in the client CLASSPATH:

- `%BEA_HOME%\integration\lib\wliclient.jar`—Contains EJB remote interfaces, home classes, and signature classes for EJBs that have published APIs.
- `%BEA_HOME%\integration\lib\wlicommon.jar`—Contains classes common to server and client.

In the preceding lines, `BEA_HOME` is the environment variable that represents the directory in which WebLogic Platform is installed.

Requirements and Recommendations

In addition to specifying the JAR files, as described in the previous section, consider the following requirements and recommendations when you deploy your client applications:

- When the client is a simple Java application, you can specify the `wliclient.jar` and `wlicommon.jar` files in the system CLASSPATH.
- When you deploy an enterprise application such that it runs in the same Java Virtual Machine (JVM) as WebLogic Integration, you must specify the JAR files in the manifest classpath.

  The J2EE specification provides the manifest `Class-Path` entry for a component to specify that it requires an auxiliary JAR of classes. When you create a JAR or WAR file for your application, you include a manifest file with a `Class-Path` element that references the required JAR files. For information about how to use this manifest file, see "WebLogic Server Application Classloading" in *BEA WebLogic Server Developers Guide*, which is available at the following URL:

  `http://e-docs.bea.com/wls/docs70/programming/classloading.html`
- Although doing so is not required, we recommend that you specify the `wliclient.jar` and `wlicommon.jar` files in the manifest `Class-Path` for a deployment scenario in which WebLogic Integration and your WebLogic Integration client application do not run in the same JVM. By listing these JAR files in the manifest Class-Path, you can ensure that both applications are self-contained.
- You must maintain the same versions of the JAR files on the server and the client.

For information about developing BPM client applications, see *Programming BPM Client Applications*.

B 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 3, "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 "Deployment Tools and Procedures" in "WebLogic Server Deployment" in *BEA WebLogic Server Developers Guide*, which is available at the following URL:

`http://e-docs.bea.com/wls/docs70/programming/deploying.html`

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 B-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 WLI_HOME\adapters\dbms\lib\BEA_WLS_DBMS_ADK.ear
-name BEA_WLS_DBMS_ADK
-targets BEA_WLS_DBMS_ADK.rar@MyCluster,
BEA_WLS_DBMS_ADK_Web.war@MyCluster,
BEA_WLS_DBMS_ADK_EventRouter.war@MyServer1
-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. (*`WLI_HOME`* represents the directory in which you installed WebLogic Integration, for example `C:\bea\weblogic700\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. (Note that there are no spaces between the items in the list.) As described in "Deploying Adapters" on page 2-30, the event router WAR files for sample adapters are targeted to a single node in a cluster. This sample command specifies that the RAR and design-time Web application are deployed to the cluster, and that the EventRouter Web application is deployed to a specific managed server.
- `-activate`—Activates the application in the domain.

Using the WebLogic Server Administration Console

1. Start the administration server and the Administration Console:

   a. To start the administration server, see "Starting WebLogic Integration" in "Getting Started" in *Starting, Stopping, and Customizing BEA WebLogic Integration*.

   b. To start the console, see "Starting the WebLogic Server Administration Console" in "WebLogic Integration Administration and Design Tools" in *Starting, Stopping, and Customizing BEA WebLogic Integration*.

2. In the Administration Console navigation tree, select the Applications node in the domain in which you want to deploy an adapter:

   Domain_Name→Deployments→Applications

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:

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

   In the preceding line, *`WLI_HOME`* represents the directory in which you installed WebLogic Integration, for example, `C:\bea\weblogic700\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 B-1.

For information about using the WebLogic Server Administration Console to deploy applications, see "Configuration and Deployment Tasks" in "Applications" in the *Adminstration Console Online Help*, which is available at the following URL:

`http://edocs.bea.com/wls/docs70/ConsoleHelp/applications.html`

Index

D

E

F

H

I

J

K

L

M

N

T

U

V

W

X