BEA WebLogic Integration™

Deploying BEA WebLogic Integration Solutions

Release 2.1
Document Date: October 2001
Revised: November 30, 2001

Deploying BEA WebLogic Integration Solutions

Part Number	Date	Software Version
N/A	Release: October 2001 Revised: November 30, 2001	2.1

Contents

About This Document

1. Introduction

2. Configuring WebLogic Integration Clusters

3. Using WebLogic Integration Security

4. Tuning Performance

Index

About This Document

This document describes how to deploy an integration solution using BEA WebLogic Integration in a production environment. 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 today's e-businesses face, with their collections of fragmented, heterogeneous business systems. It also describes the application integration, B2B integration, business process management, and data integration functionality WebLogic Integration provides 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.

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 descriptions of deployment resources, concepts, tasks, and roles on a deployment team.
- Chapter 2, "Configuring WebLogic Integration Clusters," describes how to deploy an integration solution on a cluster, which is a collection of servers that are managed as a single unit. It describes key clustering concepts and design tasks, and it provides instructions for configuring a clustered deployment.
- Chapter 3, "Using WebLogic Integration Security," describes how to set up a secure WebLogic Integration deployment.
- Chapter 4, "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 WebLogic Integration, see the following documents:

- *Introducing BEA WebLogic Integration* at the following URL:

  `http://edocs.bea.com/wlintegration/v2_1/overview/index.htm`
- *Learning to Use BEA WebLogic Integration*

  `http://edocs.bea.com/wlintegration/v2_1/tutorial/index.htm`

- *Installing BEA WebLogic Integration*

  `http://edocs.bea.com/wlintegration/v2_1/install/index.htm`

- *Starting, Stopping, and Customizing BEA WebLogic Integration*

  `http://edocs.bea.com/wlintegration/v2_1/config/index.htm`

- *BEA WebLogic Server Administration Guide* at the following URL:

  `http://edocs.bea.com/wls/docs61/adminguide/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 that you are using the documentation for the BEA WebLogic Integration 2.1 release.

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
boldface text	Indicates terms defined in the glossary.
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. *Examples*: LPT1 SIGNON OR

Convention	Item
`{ }`	Indicates a set of choices in a syntax line. The braces themselves should never be typed.
`[ ]`	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
- Deployment Architecture
- Key Deployment Resources
- Roles in Integration Solution Deployment
- Key Deployment Tasks

Deployment Goals

When deploying WebLogic Integration solutions, consider the following goals:

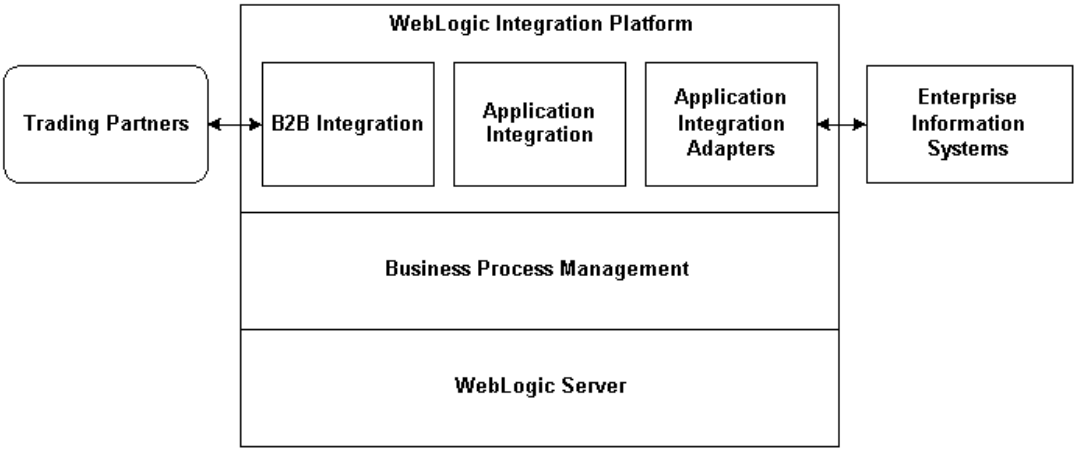
- *Performance.* A deployment must deliver sufficient performance at peak and off-peak loads.
- *Scalability.* A deployment must be capable of handling anticipated increases in loads simply by using additional hardware resources, rather than requiring code changes.
- *Security.* A deployment must sufficiently protect data from unauthorized access or tampering.

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

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 these resources in detail.

Key Deployment Resources

This section provides an overview of resources that can be modified at deployment time. It contains the following sections:

- 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 deploying 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

For more information, see the BEA WebLogic Server documentation at the following URL:

`http://edocs.bea.com/wls/docs61/index.html`

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, “Configuring 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 queues can be configured to handle workflow notifications and messages, as described in "Business Process Management Resources" on page 1-6.

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/docs61/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/docs61/adminguide/jms.html`

- "Monitoring JMS" in the "Monitoring Servers" section of "Monitoring a WebLogic Server Domain" in the *BEA WebLogic Server Administration Guide* at the following URL:

  `http://edocs.bea.com/wls/docs61/adminguide/monitoring.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 event listener message-driven beans, `max-beans-in-free-pool` is the maximum number of listeners that pull work from a queue.

- For stateless session beans, `max-beans-in-free-pool` is the maximum number of beans available for work requests.

- For stateful session beans and entity beans, `max-beans-in-cache` is 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.

The WebLogic Server documentation recommends setting the number of execute threads rather than setting `max-beans-in-free-pool`. In a WebLogic Integration environment, however, 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.

For more information about configuring pool and cache sizes, see "Configuring Other EJB Pool and Cache Sizes" on page 4-3.

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. 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 the following sections:

- "Overview of Connection Pools" in "Introduction to WebLogic JDBC" in *Programming WebLogic JDBC* at the following URL:

  `http://edocs.bea.com/wls/docs61/jdbc/intro.html`

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

  `http://edocs.bea.com/wls/docs61/adminguide/jdbc.html`

- "Monitoring JDBC Connection Pools" in "Monitoring a WebLogic Server Domain" in the *BEA WebLogic Server Administration Guide* at the following URL:

  `http://edocs.bea.com/wls/docs61/adminguide/monitoring.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.

In a WebLogic Integration environment, controlling the number of message-driven beans is generally the best way to throttle work. The execution thread pool should be set sufficiently high for all candidate threads to run. In general, a pool size that is too large is preferable to one that is too small.

For more information about configuring the execution thread pool, see "Configuring the Execution Thread Pool" on page 4-6.

J2EE Connector Architecture

The WebLogic J2EE Connector Architecture (J2EE-CA) 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 more 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/docs61/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

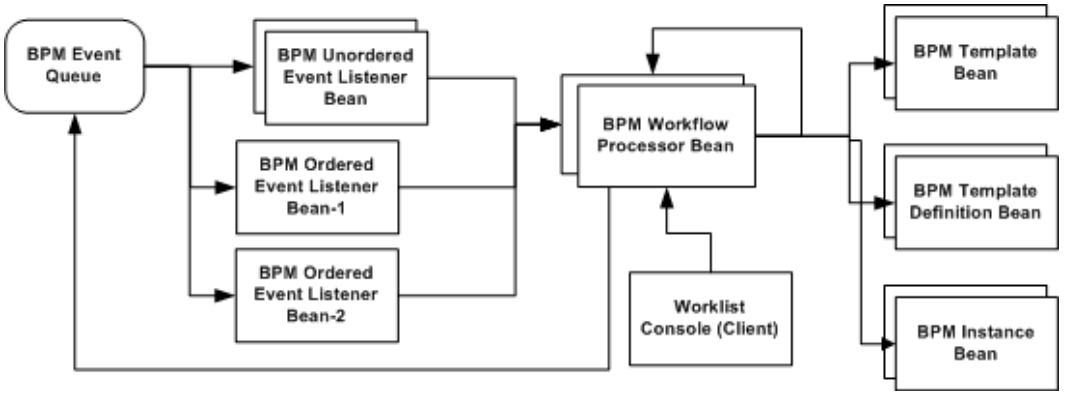
- BPM Work Sequence

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



For a description of these resources, see the next section, "Types of BPM Resources."

Types of BPM Resources

BPM uses WebLogic JMS (described in "Java Message Service" on page 1-4) 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 create custom message queues using the WebLogic Server Administration Console, run the MDBGenerator utility to generate an event listener bean to listen on the queue, and update the BPM configuration to recognize the new event listener bean. For more information, see "Configuring JMS Servers and Queues for Application Integration" on page 2-25.

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

Workflow Processor Beans

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

Workflow processor beans are instantiated at run time based on the system load, so 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 container in the pool becomes available. In general, a pool size that is too large is preferable to one that is too small.

Workflow processor beans are clusterable (they have cluster-aware stubs), so they may accept work from other nodes in the cluster.

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 execute. 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

The event queue is a JMS queue that is tied to a specific JMS server. WebLogic Integration ships with a default event queue in the domain configurations. You can also create new event queues, as described in "Configuring JMS Queues for BPM" on page 2-24.

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

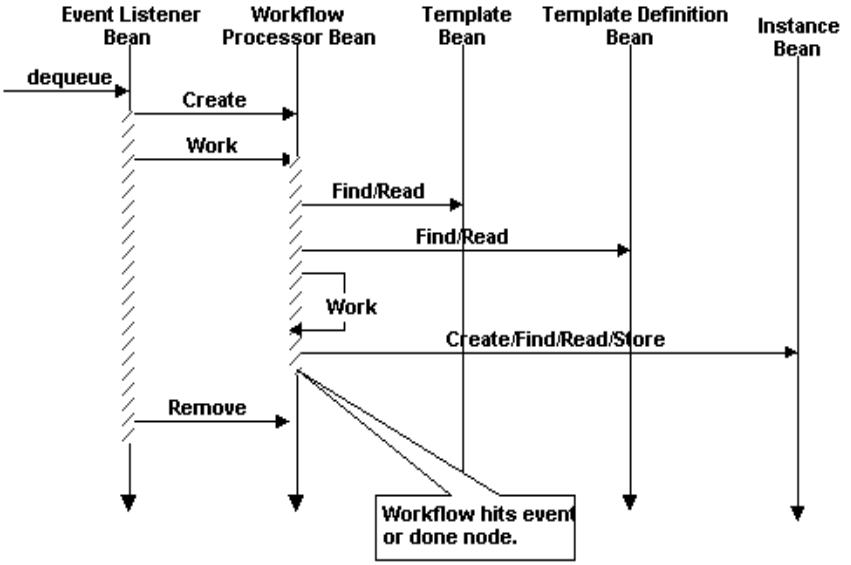
Worklist Console

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

B2B integration resources are allocated dynamically, as needed; the 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*.

While you cannot configure WebLogic Server resources to configure B2B integration, you should be aware that some B2B integration resources, such as delivery channels for hub-and-spoke configurations, affect performance. For more information, see "XOCP Hub and Spoke Delivery Channels" in "Configuration Requirements" in *Administering B2B Integration*.

B2B integration functionality can be deployed on a WebLogic Integration cluster, but resources must be targeted to a single node in the cluster. For more information about clustering and B2B integration, see Chapter 2, "Configuring WebLogic Integration Clusters."

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 integration can be configured for a group of servers that are managed as a single unit. For more information about clustering and application integration, see Chapter 2, "Configuring WebLogic Integration Clusters."

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

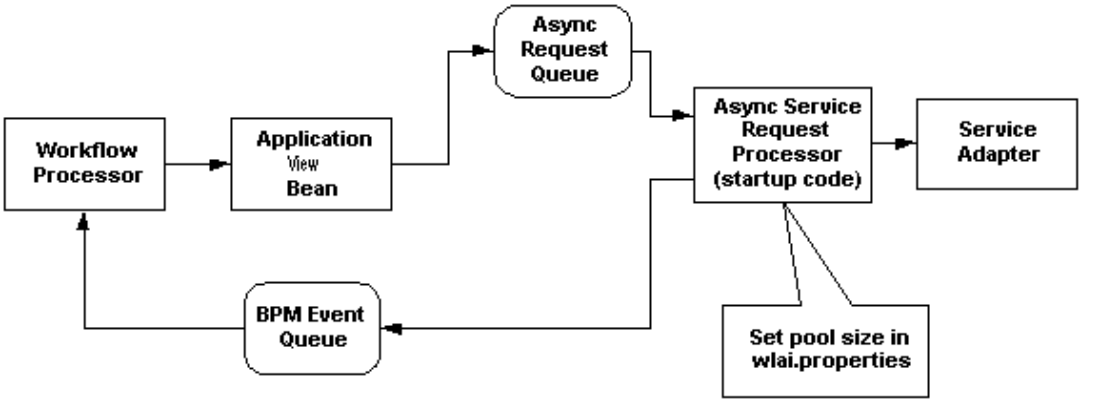
In a synchronous service invocation, a client (shown here as a *workflow processor*) calls the *application view EJB* (a stateless session bean). The application view calls the *service adapter* using a synchronous CCI request. The service adapter is a J2EE-CA service adapter that actually processes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

Figure 1-5 Asynchronous Service Invocations



In an asynchronous invocation, a client (shown here as a *workflow processor*) calls the *application view EJB* (a stateless session bean). The application view EJB then queues the request for later processing and returns control to the client. The asynchronous request is queued in the *async request queue* (`wlai.myserver.ASYNC_REQUEST` in the default domain). This is a JMS queue in a JMS server.

An *async service request processor* pulls work from the async request queue, calls the service adapter, and enqueues the response in the *async response queue*. The async service request processor is configured as a pool of JMS listeners in the `wlai` startup class. The pool size can be configured by a property in the `wlai.properties` file (`wlai.numAsyncServiceRequestProcessors`). Setting the pool size of the async service request processor is the best way to control the amount of work that a service adapter can perform when using asynchronous service invocations.

Events

The following figure illustrates the information flow of an event.

Figure 1-6 Events

An *event adapter* generates events and places them in a *temporary topic* created at run time. The temporary topic is a JMS Topic associated with a JMS Server. The events are forwarded to an *EngineMgr* that is listening on the temporary topics. The EngineMgr is a JMS listener that is initiated by the EngineMgr EJB.

A subscribing application, such as a workflow, receives the forwarded event on a JMS queue. In a workflow, the application integration plug-in determines the queue on which the event should be received for a start or event node—it need not be the default BPM EventQueue. The *workflow processor* represents a client application.

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 4-33.

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 "Tuning Hardware, Operating System, and Network Resources" on page 4-29.

Roles in Integration Solution Deployment

The deployment team for an integration solution fulfills the following roles:

- Deployment Specialists
- WebLogic Server Administrators
- Database Administrators

A successful deployment requires input from *all* of these participants. Note that one person can assume multiple roles and that not all roles are equally relevant in all deployments.

Deployment Specialists

Deployment specialists coordinate the deployment effort. They are knowledgeable about the features and capabilities 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 experience in the following areas:

- Hardware and platform knowledge

- Experience in managing all aspects of a WebLogic Server deployment, including installation, configuration, monitoring, security, performance tuning, troubleshooting, and other administrative tasks

Database Administrators

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

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

Key Deployment Tasks

Deploying WebLogic Integration may require 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. Become familiar with the WebLogic Integration resources that you can use to configure a deployment, as described in "Key Deployment Resources" on page 1-2.
3. To deploy WebLogic Integration on a cluster, you must first design and then configure the cluster, as described in Chapter 2, "Configuring WebLogic Integration Clusters."
4. Set up security for your WebLogic Integration deployment, as described in Chapter 3, "Using WebLogic Integration Security."
5. Once your WebLogic Integration deployment is up and running, you can optimize overall system performance, as described in Chapter 4, "Tuning Performance."

CHAPTER

2 Configuring WebLogic Integration Clusters

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

- Understanding WebLogic Integration Clusters
- Designing a Clustered Deployment
- Configuring a Clustered Deployment

Understanding WebLogic Integration Clusters

The following sections describe clustering in WebLogic Integration deployments:

- About WebLogic Integration Clusters
- WebLogic Integration Deployment Resources
- Load Balancing in a WebLogic Integration Cluster

About 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 one and only one administration server, and one or more managed servers. The managed servers in a domain can be organized into clusters. When you configure WebLogic Integration clusterable resources, you can target selected servers or selected clusters. Targeting clusters provides the most flexibility because servers can be added to them, dynamically, to increase capacity.

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/docs61/cluster/index.html`

- "Configuring WebLogic Servers and Clusters" in the *BEA WebLogic Server Administration Guide* at the following URL:

  `http://edocs.bea.com/wls/docs61/adminguide/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/docs61/perform/WLSTuning.html`

Designing a Clustered Deployment

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

- WebLogic Integration Deployment Resources
- Load Balancing in a WebLogic Integration Cluster

Before you begin designing the architecture for your clustered domain, you need to learn how WebLogic Server clusters operate. For general information, see *Using WebLogic Server Clusters* in the WebLogic Server documentation set, at the following URL:

`http://edocs.bea.com/wls/docs61/cluster/index.html`

For details about basic, multi-tiered, and proxy architectures that are recommended, see "Planning WebLogic Server Clusters" in *Using WebLogic Server Clusters*. To learn how WebLogic Integration resources can be partitioned and distributed within a clustered domain, see "WebLogic Integration Deployment Resources" on page 2-3.

WebLogic Integration Deployment Resources

The following sections describe WebLogic Integration resources that can be deployed in a cluster:

- Resource Groups
- Deployment Containers

Resource Groups

Resource groups are collections of related deployment resources that are categorized for clustering purposes. Each resource in a resource group must be targeted to the same machine.

Types of Resource Groups

There are two types of resource groups:

- *Clusterable*—Located on one or more servers in a cluster. Certain resources must be located on all servers in the cluster.
- *Single Node*—Targeted to one and only one server in a cluster.

List of Resource Groups

The following table describes the resource groups supported by WebLogic Integration.

Table 2-1 Resource Groups for WebLogic Integration Resources

Group Name	Description	Type of Resource Group
b2b	B2B integration components.	Single node Must reside on the same node as bpm-singleNode components.
bpm-singleNode	BPM master components.	Single node
bpm-queue-default	BPM default event queue and message-driven beans.	Single node
bpm-queue-*xxx*	BPM user-defined event queue and message-driven beans.	Single node While multiple user-defined queues and message-driven beans can exist, each one can be targeted to only a single node in a cluster.
bpm-clusterable	BPM clusterable components.	Clusterable Must reside on any bpm-singleNode and any bpm-queue-default or bpm-queue-*xxx* node.
wlai-admin	Application integration administration components.	Single node
wlai-clusterable	Application integration clusterable components.	Clusterable

Table 2-1 Resource Groups for WebLogic Integration Resources (Continued)

Group Name	Description	Type of Resource Group
wlai-event-*yyy*	Application integration event adapter. **Note:** 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.	Depends on the adapter For example, the mail and DBMS sample event adapters, as well as the binary file event adapter, are all single node. For more information, see the documentation for the adapter you are using.
wlai-service-*yyy*	Application integration service adapter. **Note:** 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.	Depends on the adapter For example, the mail and DBMS sample event adapters, as well as the binary file event adapter, are all single node. For more information, see the documentation for the adapter you are using.
wlai-queue-default	Default queue for application integration.	Single node
wlai-queue-*zzz*	Application integration queue for a specific server.	Single node
wli-clusterable	Resources that must be located on all servers in the domain.	Clusterable

Deployment Containers

The table provided in this section describes the WebLogic Integration deployment resources. These resources can be viewed and modified in the WebLogic Server Administration Console. The table provides the following information for each package or service:

- *Resource Group*—Arbitrary designation for a set of related resources that must be deployed together.
  - All singleNode resources must be targeted to one, and only one, server. They cannot be clustered.
  - Clusterable resources can be targeted to one or more servers, but all members of a group must be targeted to the same server or set of servers.

- *Resource Name*—Name of an individual package or service as it is shown in the WebLogic Server Console. 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
- *Container*—Route through the WebLogic Server Administration Console navigation tree to the specified package or service.

Note: Names shown in italic represent user-defined packages or resource groups that are not part of the default domain.

Table 2-2 WebLogic Integration Deployment Containers

Resource Group	Resource Name	Container
b2b	Wlcconsole	Deployments→Application→WLC console
	WLCShutdown	Deployments→Shutdown
	WLCStartup	Deployments→Startup
	WLCHub.DS	Services→JMS→Tx Data Sources
bpm-singleNode	Time processor	Deployments→Application→WLPI Application
	Wlpi-master-ejb	Deployments→Application→WLPI Application
	WLPIInit	Deployments→Startup
bpm-queue-default	Wlpi-mdb	Deployments→Application→WLPI Application
	JMSServer-0	Services→JMS/Servers
bpm-queue-xxx	*Wlpi-mdb-xxx*	Deployments→Application→WLPI Application
	Wlpi-queue-xxx	Deployments→Application→WLPI Application

Table 2-2 WebLogic Integration Deployment Containers (Continued)

Resource Group	Resource Name	Container
bpm-clusterable	Wlc-wlpi-plug-in	Deployments→Application→WLPI Application
	Respository-ejb	Deployments→Application→WLPI Application
	Wlpi-ejb	Deployments→Application→WLPI Application
	Wlxtejb	Deployments→Application→WLPI Application
	Wlxtpi	Deployments→Application→WLPI Application
	WLXTPlugin	Deployments→Application→WLPI Application
	WLPIFactory	Services→JMS→Connection Factories
	WLPIQueueFactory	Services→JMS→Connection Factories
	TxDataSource	Services→JMS→Tx Data Sources
wlai-admin	Wlai-admin-ejb	Deployments→Application→WLAI Application
wlai-clusterable	Wlai	Deployments→Application→WLAI Application
	Wlai-ejb-server	Deployments→Application→WLAI Application
	WlaiStartup	Deployments→Startup
wlai-event-yyy	*yyyEventRouter*	Deployments→Application→*yyy*EventRouter **Note:** 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.

Table 2-2 WebLogic Integration Deployment Containers (Continued)

Resource Group	Resource Name	Container
wlai-service-yyy	*BEA . . . yyy . . . ADK_RAR*	Deployments→Application→ BEA . . . *yyy* . . . ADK_RAR **Note:** 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.
	BEA . . . yyy . . . ADK_WEB	Deployments→Application→ BEA . . . yyy . . . ADK_WEB
wlai-queue-default	Wlai_JMSServer	Services→JMS→Servers
wlai-queue-zzz	*WLAI_zzz_ JMSServer*	Services→JMS→Servers
wli-clusterable	*Mailsession*	Java mail sessions used for the BPM Send E-mail action.
	JDBCConnectionPool	Used for all database connections in WebLogic Integration.
	JMSConnectionFactory	Used for all JMS connections in WebLogic Integration.
	JDBCTtxDataSource	Transaction coordinator pool used for all transactions in WebLogic Integration.

Load Balancing in a WebLogic Integration Cluster

Load balancing distributes the workload proportionally among all the servers in a cluster so that each server can each run at full capacity. The following sections describe load balancing in a WebLogic Integration cluster:

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

Load Balancing WebLogic Server in a Cluster

WebLogic Server supports load balancing for HTTP session states and clustered objects. For more information, see *Using WebLogic Server Clusters* in the WebLogic Server documentation set, at the following URL:

`http://edocs.bea.com/wls/docs61/cluster/index.html`

Load Balancing BPM in a Cluster

BPM workflows require an event queue for processing event-based workflows. For more information, see "Business Process Management Resources" on page 1-6.

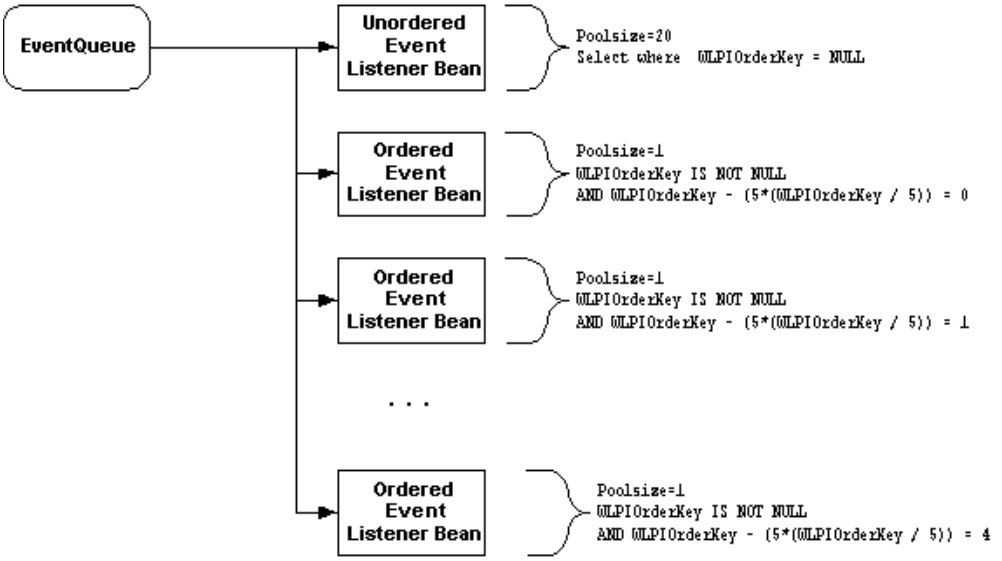
Event Queues and Associated Pools

Two 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`.

The message producer is responsible for delivering the messages in the queue in the correct order.

Note: `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. If you are using messages between workflows, you can set `WLPIOrderKey` in the post-XML event dialog box. For more information, see "Posting an XML Message to a JMS Topic or Queue" in "Defining Actions" in *Using the WebLogic Integration Studio*.

A single jar file contains both ordered and unordered event listener message-driven beans for a particular queue. The WebLogic Integration installation provides the `wli-mdb-ejb.jar` file to be read from the default EventQueue. To ensure that the ordered event listener message-driven beans preserve processing order, this jar file must be targeted to one—and only one—server in a cluster.

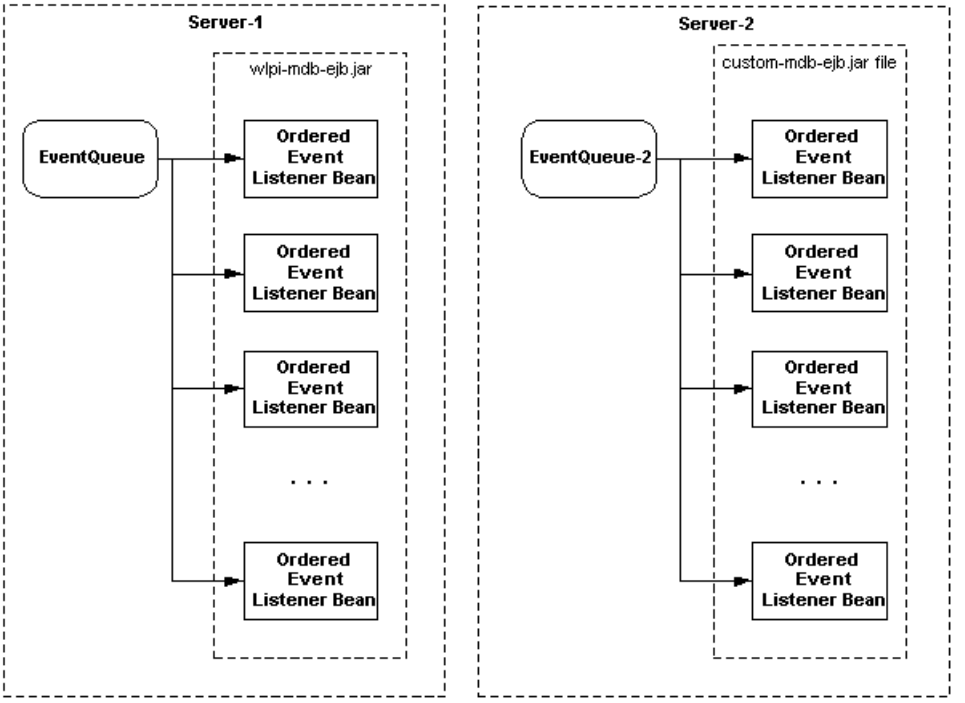
Creating New Pools

If you have sufficient processing power on a single server, you can increase the pool size for event listener message-driven beans in the `wli-mdb-ejb.jar` file. If you want to put more event listener message-driven beans on another server, you need to create a new jar file and a new queue.

The pool size and range of `WLPIOrderKeys` can be changed for `wli-mdb-ejb.jar`, but they *cannot* be targeted to another server. To add event listener message-driven beans on another server, you must create a new pool using the MDBGenerator utility. For more information, see "Configuring a Custom Java Message Service Queue" in "Customizing WebLogic Integration" in *Starting, Stopping, and Customizing BEA WebLogic Integration*.

The following diagram illustrates the default queue (EventQueue) on Server-1 and a user-defined queue (EventQueue-2) on Server-2. The new pool targets a different queue (EventQueue-2), and that queue must be shared with another event listener message-driven bean pool. In this diagram, the custom-generated event listener message-driven beans pull work from the user-defined event queue.

Figure 2-2 Event Listener Message-Driven Beans on Another Server



An application *must* send work to the user-defined queue in order to use the capacity of the custom generated event listener message-driven beans.

Requirements for Load Balancing BPM Functionality

When load balancing BPM functionality in a WebLogic Integration cluster, you must comply with 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 one and only one server in a cluster. By default, WebLogic Integration provides the `wlpi-mdb-ejb.jar` file to pull work from `com.bea.wlpi.EventQueue`.

  Note: This also applies to the validating queue when using XML validation. The default validating queue is `com.bea.wlpi.ValidatingEventQueue`.

- Use the MDBGenerator 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.

Load Balancing Application Integration in a Cluster

Application integration does not require partitioning of work within a cluster. It is possible to configure a completely homogenous cluster (that is, one in which all resources have the same 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. New JMS servers are needed only when the default JMS server is saturated.

Configuring a Clustered Deployment

The following sections describe how to configure a clustered WebLogic Integration deployment:

- Configuration Prerequisites
- Summary of Basic Configuration Tasks
- Setting Up a WebLogic Integration Managed Server
- Adding Managed Servers and Creating Clusters
- Configuring JMS Queues for BPM
- Configuring JMS Servers and Queues for Application Integration
- Distributing Resources Across Servers or Clusters
- Starting the Servers in the Domain
- Load Balancing in a WebLogic Integration Cluster

Configuration Prerequisites

Before configuring WebLogic Integration to run in a clustered environment, you must:

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

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

- Have a static IP address for each participant in the clustered domain.

  Each WebLogic Server instance requires a unique, static IP address. If multiple servers are run on a single machine, that machine must be configured as a multihomed server. Obtain an IP address for each WebLogic Server instance to be started on each machine in the cluster.

- Have a multicast address for each cluster.

  Clustered servers communicate among themselves over multicast and must share a single, exclusive multicast address.

- Assign the listen port that will be used by all servers in the domain.

  All servers in a domain must be configured with the same listen port. If you have already assigned a listen port to the administration server, you must assign the same port to each managed server.

Note: There are additional requirements when the architecture of the domain includes one or more firewalls. For more information, see *Using WebLogic Server Clusters* in the WebLogic Server documentation set, at the following URL:

`http://edocs.bea.com/wls/docs61/cluster/index.html`

Summary of Basic Configuration Tasks

Configuring WebLogic Integration in a clustered environment involves the following basic tasks:

1. Planning the architecture of the clustered domain, as described in "Designing a Clustered Deployment" on page 2-3.
2. Setting up the required WebLogic Integration managed servers, as described in "Setting Up a WebLogic Integration Managed Server" on page 2-15.
3. Adding a definition for each managed server to the domain configuration and assigning all managed servers to clusters, as described in "Adding Managed Servers and Creating Clusters" on page 2-22.
4. Deploying application resources across servers or clusters, as described in "Distributing Resources Across Servers or Clusters" on page 2-26.

The instructions are based on two assumptions:

- You have configured an Oracle, Microsoft SQL, or Sybase database for an existing domain.
- You will add servers to that domain.

Setting Up a WebLogic Integration Managed Server

The following sections provide instructions for setting up a managed server in two situations:

- Adding a Managed Server to the Existing Installation
- Installing a Managed Server in a New Location

The information you need for both of these procedures is provided in the following sections. For simplicity, it is assumed that you are adding a managed server to one of the preconfigured domains. If you have created your own domain, the process is similar.

Adding a Managed Server to the Existing Installation

Before multiple instances of WebLogic Server can be supported on the same machine, the machine must be configured as a multihomed host. If you meet this requirement, you can add a managed server to a domain in the existing installation, simply by adding a command file that starts the managed server to the domain directory.

Note: If you are adding a managed server to a domain in which adapters, applications views, and the application integration plug-in are deployed, then the *WLI_HOME*\config*domain_name* directory must also contain the following:

WLI_HOME\config\domain_name\wlai\wlai.*managed_svrname*.properties
WLI_HOME\config\domain_name\wlai\deploy*.rar

where *.rar represents any adapters that are available for deployment.

For example, to add a managed server to the WebLogic Integration (config\wlidomain) domain in the existing installation:

1. Make a copy of the start server command file by entering the command that is appropriate for your platform:
   - Windows:

     copy *WLI_HOME*\config\wlidomain\startWeblogic.cmd
     WLI_HOME\config\wlidomain\startManagedWeblogic.cmd
   - UNIX:

     cp *WLI_HOME*/config/wlidomain/startWeblogic.sh
     WLI_HOME/config/wlidomain/startManagedWeblogic.sh
2. Open the new startManagedWeblogic command file in your preferred text editor.
3. Add the following option to the Java command that starts the server:

   -Dweblogic.management.server=*virtualhost*:7001

   Here, *virtualhost* is the IP address or DNS name assigned to the administration server on the machine, and 7001 is the assigned listen port.
4. Edit the -Dweblogic.Name option to assign a name to the managed server as follows:

`-Dweblogic.Name=`*`managed_svrname`*

For an example of a modified server start command file, see "Sample Managed Server Startup" on page 2-21.

5. Save and close the file.
6. If you are adding a managed server to a domain in which adapters, applications views, and the application integration plug-in are deployed, then complete the following steps:
   a. Go to the *`WLI_HOME`*`\config\wlidomain\wlai` directory on a Windows system, or to the equivalent on a Unix system.
   b. Copy the `wlai.properties` file to `wlai.`*`managed_svrname`*`.properties`.

      Here, *`managed_svrname`* is the name of the managed server.
   c. Open the new `wlai.`*`managed_svrname`*`.properties` file in your preferred text editor.
   d. Add the following parameters to the file:

      ```
      wlai.admin.hostNameandPort=virtualhost:7001
      wlai.admin.hostUserID=user
      wlai.admin.Password=passwd
      wlai.numAsyncServiceRequestProcessors=4
      ```

      Here, *`virtualhost`* is the IP address or DNS name of the administration server, `7001` is the port assigned to the `wlidomain` administration server, `user` is a valid user ID (typically `system`), and `passwd` is a valid password for the user.
   e. Modify the following properties:

      ```
      wlai.jms.serverName=managedsvr_jmsserver
      wlai.jms.connectionFactoryJNDIName=connection_factory_name
      ```

      Here, *`managedsvr_jmsserver`* is the JMS server for the managed server, and *`connection_factory_name`* is the name of the connection factory, if a separate JMS connection factory is required for the server.
   f. Save and close the file.
   g. Add the following line to the `fileRealm.properties` file:

      ```
      acl.access.weblogic.admin.mbean=everyone
      ```

Note: In the `startManagedWeblogic` file that results from the preceding procedure, some unnecessary environment variables are set, but they do not interfere with the execution of the command.

Before you can start the new managed server, you must add it to the configuration for the domain, as described in "Adding Managed Servers and Creating Clusters" on page 2-22.

Installing a Managed Server in a New Location

The simplest way to set up a managed server in a new location is to:

- Install WebLogic Server and WebLogic Integration in the new location.
- Modify the contents of a preconfigured domain directory to serve as the start location for the managed server.

At a minimum, you must include the following in the domain directory for the managed server:

- A command file to start the managed server must be located in the *`WLI_HOME`*`\config\`*`domain_name`* directory in the new installation.

  For example, the pathname for your start command file might be *`WLI_HOME`*`\config\ABC_Systems\startWeblogic`.

- The `setWliDomainData` command file

  Each preconfigured domain contains a `SetXxxDomainData` command. This file is specific to the preconfigured domains, the domain classpath, and other variables. This file might not exist if you have created your own domain.

- The value of *`domain_name`* must be the same as the *`domain_name`* value for all other servers in the domain.

If you are adding a managed server to a domain in which adapters, application views, and the application integration plug-in are deployed, the `config\`*`domain_name`* directory must also contain the following:

`WLI_HOME``\config\domain_name\wlai\wlai.`*`managed_svrname`*`.properties`
`WLI_HOME``\config\domain_name\wlai\deploy\*.rar`

where `*.rar` represents any adapters that are available for deployment.

For example, to add a managed server to the WebLogic Integration (*WLI_HOME*\config\wlidomain) domain:

1. Install WebLogic Server and WebLogic Integration on the machine that will host the managed server.
2. Delete any unnecessary files and directories from the *WLI_HOME*\config\wlidomain directory.

   Note: The following files *are* necessary and must *not* be deleted:

   - *WLI_HOME*\config\wlidomain\startWeblogic.cmd or startWebLogic.sh
   - *WLI_HOME*\config\wlidomain\SetWliDomainData.cmd or SetWliDomainData.sh
   - *WLI_HOME*\config\wlidomain\wlai\wlai.properties
   - *WLI_HOME*\config\wlidomain\wlai\deploy*MyResourceAdapter**.rar

   Here, *MyResourceAdapter**.rar represents any resource adapters that are available for deployment.
3. Open the startWebLogic command file in your preferred text editor.
4. Add the following option to the Java command that starts the server:

   -Dweblogic.management.server=*host:*7001

   Here, host is the IP address or DNS name assigned to the administration server, and 7001 is the assigned listen port.
5. Edit the -Dweblogic.Name option to assign a name to the managed server, as follows:

   -Dweblogic.Name=*managed_svrname*

   For an example of a modified server start command file, see "Sample Managed Server Startup" on page 2-21.
6. Save the file as startManagedWeblogic and close it. Retain the original startWeblogic command file. Once you successfully start the managed server using the startManagedWeblogic command file, you can delete the original.
7. Open the SetWliDomainData command file.
8. Locate the following lines:

```
REM load database specific data
call %WLI_HOME%\config\wlidomain\setdbdata || goto error
```

9. Comment out the following line:

   ```
   REM call %WLI_HOME%\config\wlidomain\setdbdata || goto error
   ```

10. Save and close the file.

11. If you are adding a managed server to a domain in which adapters, applications views, and the application integration plug-in are deployed, then complete the following steps:

    a. Rename the *WLI_HOME*\config\wlidomain\wlai\wlai.properties file as *WLI_HOME*\config\wlidomain\wlai\wlai.*managed_svrname*.properties.

       Here, *managed_svrname* is the name of the managed server.

    b. Open the wlai.*managed_svrname*.properties file in your preferred text editor.

    c. Add the following parameters to the file:

       ```
       wlai.admin.hostNameandPort=host:7001
       wlai.admin.hostUserID=user
       wlai.admin.Password=passwd
       wlai.numAsyncServiceRequestProcessors=4
       ```

       Here, *host* is the IP address or DNS name of the administration server, 7001 is the port assigned to the wlidomain administration server, *user* is a valid user ID (typically system), and *passwd* is a valid password for the user.

    d. Modify the following properties:

       ```
       wlai.jms.serverName=managedsvr_jmsserver
       wlai.jms.connectionFactoryJNDIName=connection_factory_name
       ```

       Here, *managedsvr_jmsserver* is the JMS server for the managed server, and *connection_factory_name* is the name of the connection factory, if you require a separate connection factory for this server.

    e. Save and close the file.

    f. Add the following line to the fileRealm.properties file:

       ```
       acl.access.weblogic.admin.mbean=everyone
       ```

Note: In the `startManagedWeblogic` file that results from the preceding procedure, some unnecessary environment variables are set, but they do not interfere with the execution of the command.

Before you can start the new managed server, you must add it to the configuration for the domain, as described in "Adding Managed Servers and Creating Clusters" on page 2-22.

Sample Managed Server Startup

The following code listing shows an example of the start server command for a `wlidomain` managed server. The modifications from the *`WLI_HOME`*`\config\wlidomain\startWebLogic` command for the default administration server are shown in bold. This code listing represents a single command. It is shown here on multiple lines for the sake of readability. In your command file, however, it is entered as one physical line.

Listing 2-1 Start Server Command for a wlidomain Managed Server

```
%JAVA_HOME%\bin\java %DB_JVMARGS% -Xmx256m -classpath %SVRCP%
 -Dbea.home=%BEA_HOME% -Dweblogic.home=%WL_HOME%
 -Dweblogic.system.home=%WLI_HOME% -Dweblogic.Domain=wlidomain
 -Dweblogic.management.password=security
 -Dweblogic.management.server=172.20.50.250:7001
 -Dweblogic.Name=managed1 -Dweblogic.RootDirectory=%WLI_HOME%
 -Djava.security.policy=%WL_HOME%\lib\weblogic.policy
 -Dweblogic.management.discover=false weblogic.Server
```

Adding Managed Servers and Creating Clusters

When you add a managed server to a domain, you perform the following tasks:

1. Create the required clusters according to the instructions in “Creating a Cluster” on page 2-22.
2. If the domain includes more than one machine, create a machine definition for each machine that runs a server instance. See “Creating a Machine” on page 2-23. Any servers that existed before you created machines must be assigned to a machine. See “Assigning an Existing Server to a Machine or Cluster” on page 2-24.
3. Create the required servers according to the instructions in “Creating a Server” on page 2-23. As part of each server definition, assign the server to both a cluster and a machine.

All tasks are performed 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*.

Creating a Cluster

To create a new cluster:

1. In the navigation tree, select Clusters to display the Clusters page.
2. Click the Configure a New Cluster link.
3. On the General tab, enter values for the Name (arbitrary identifier) and Cluster Address (multicast address) fields.

   Note: The valid range of IP addresses for multicast is between 224.0.0.1 and 239.255.255.255. The address must *not* be used by any other application in the subnet.
4. Click Create.
5. Set other options as required, and click Apply when finished.

   The defaults are usually acceptable for an initial configuration.

Creating a Machine

To create a machine:

1. In the navigation tree, select Machines.
2. Click the Configure a New Machine link.
3. Enter the name that will be used to identify the machine. Any arbitrary identifier can be used.
4. Click Create.

Creating a Server

To create a server:

1. In the navigation tree, select Servers.
2. Click the Configure a New Server link.
3. 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.

   Note: The value of Name must match the name assigned, in the start command, to the managed server. See "Setting Up a WebLogic Integration Managed Server" on page 2-15.
4. Select the machine name from the Machine drop-down list.
5. Click Create.
6. Select the Cluster tab.
7. Select the appropriate cluster from the Cluster drop-down list.

Assigning an Existing Server to a Machine or Cluster

To assign an existing server to a machine:

1. In the navigation tree, select the name of the machine to which you will assign the server.
2. Select the Servers tab.
3. Move the server or servers to be assigned to the machine from the Available list to the Chosen list.
4. Click Apply.

To assign an existing server to a cluster:

1. In the navigation tree, select the name of the cluster to which you will assign the server.
2. Select the Servers tab.
3. Move the server or servers to be assigned to the machine from the Available list to the Chosen list.
4. Click Apply.

Configuring JMS Queues for BPM

After adding servers, you need to add JMS queues for BPM. The BPM event listener message-driven bean pool associated with a particular JMS queue must be collocated with the queue's JMS server. For instructions, see "Configuring a Custom Java Message Service Queue" in "Customizing WebLogic Integration" in *Starting, Stopping, and Customizing BEA WebLogic Integration*.

Configuring JMS Servers and Queues for Application Integration

This section describes how to configure JMS servers and queues for application integration in a clustered environment. To determine how many JMS servers are required, see "Load Balancing Application Integration in a Cluster" on page 2-13.

Create a Store and Associate It with a Connection Pool

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

1. In the navigation tree, go to the Services→JMS→Stores node and select Create 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 "Create a JMS Server and Associate It with the Store" on page 2-25.
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, WLAI).
5. Click Create.

Create a JMS Server and Associate It with the Store

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

1. In the navigation tree, go to the Services→JMS→Servers node and select Create a new JMSServer.
2. In the Name field, enter a name by which you want to identify this JMS server.

   For JMS servers for BPM, the JMS server name must match the name used when generating the message-driven bean jar file using the MDBGenerator utility.
3. In the Store field, select the JMSJDBCStore with which you want to associate this JMS server.

4. In the Temporary Template field, select WLAI_TemporaryTopicTemplate.
5. Click Create.

Distributing Resources Across Servers or Clusters

After you set up the managed servers to be included in the domain, add definitions for those servers to the configuration, and assign the servers to clusters as required, you are ready to modify the WebLogic Integration resource configuration.

Distribution Guidelines

Before proceeding with this step, make sure you have outlined a deployment plan that conforms to the following guidelines:

- Certain resources must be deployed to all servers in the domain. For more information, see “Deployment Containers” on page 2-5.
- Certain resources must be deployed to one and only one server in the domain. For more information, see “Deployment Containers” on page 2-5.
- The number of JMS servers and queues should be determined using the guidelines described in “Load Balancing BPM in a Cluster” on page 2-9 and “Load Balancing Application Integration in a Cluster” on page 2-13.
- The administration server doe not require WebLogic Integration resources.
- Resources identified as members of the same resource group must be targeted to the same server, or, if clusterable, to the same set of servers, as described in “WebLogic Integration Deployment Resources” on page 2-3.

Once you have a plan for deploying your resources to the servers and clusters in your domain, the procedure for modifying the resource configuration depends on the resource.

Targeting Resources to Clusters

To target a resource to, or remove a resource from, particular clusters:

1. In the navigation tree, select the resource.
2. Select the Targets tab.
3. Select the Clusters tab.
4. Modify the chosen Targets—Clusters as required.
   - To target a resource to a cluster, move the cluster from the Available list to the Chosen list.
   - To remove a resource from a cluster, move the cluster from the Chosen list to the Available list.
5. Click Apply when you have finished making changes.

Targeting Resources to Servers

To target a resource to, or remove a resource from, a particular server (for example, the administration server):

1. In the navigation tree, select the resource.
2. Select the Targets tab.
3. Select the Servers tab.
4. Modify the chosen Targets—Servers as required.
   - To target the resource to a cluster, move the cluster from the Available list to the Chosen list.
   - To remove a resource from a cluster, move the cluster from the Chosen list to the Available list.
5. Click Apply when you have finished making changes.

Starting the Servers in the Domain

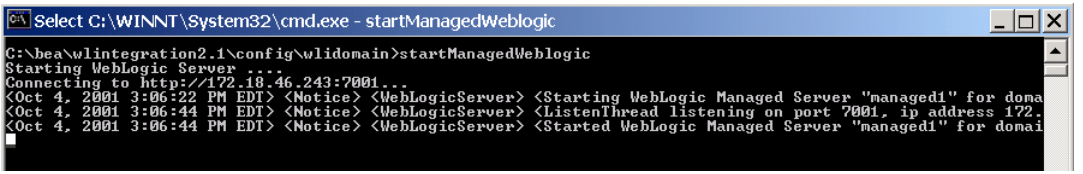
After you configure the servers in your domain, you can bring them up in the following order:

1. If it is not already running, start the administration server.
2. Execute the `startManagedWeblogic` command for each managed server. For example, if you installed a managed server for the `wlidomain` on an NT machine, enter the following command:

   ```
   cd bea\wlintegration2.1\config\wlidomain
   startManagedWeblogic
   ```

As the managed server starts, the follow messages are displayed.

Figure 2-3 Managed Server Startup Messages



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

When you are ready to shutdown the server, you can use the WebLogic Server Administration Console as well.

CHAPTER

3 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
- WebLogic Server Security Principals Used in WebLogic Integration

Overview of WebLogic Integration Security

The following sections describe the key features of WebLogic Integration security:

- WebLogic Server Security
- Business Process Management Security
- B2B Integration Security
- Application Integration Security

Every secure deployment of a WebLogic Integration solution uses the WebLogic Server security features as the foundation. If BPM, B2B integration, and application integration functionality is used in a WebLogic Integration solution, the security features associated with these functional areas are used as well.

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 collocated applications.

WebLogic Server Security

WebLogic Server provides the foundation for WebLogic Integration security. WebLogic Integration deployments can use a full range of security features that WebLogic Server provides, including security realms, users and groups, Access Control Lists (ACLs) and permissions, the Secure Sockets Layer (SSL) protocol, authentication mechanisms, digital certificates, controlled access to resources, and so on. For a comprehensive discussion of WebLogic Server security features, see "Programming WebLogic Security" at the following URL:

`http://edocs.bea.com/wls/docs61/security/index.html`

Business Process Management Security

WebLogic Integration uses WebLogic Server security realms to protect access to workflows and other resources. User access is determined by the roles to which the user is assigned. The WebLogic Integration Studio is used to define users, organizations, and roles, and also to map roles to groups in WebLogic Server security realms.

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

- "About Security Realms" in "Administering Data" in *Using the WebLogic Integration Studio*
- "Configuring the Security Realms" in *Programming BPM Client Applications*

B2B Integration Security

WebLogic Integration solutions that involve the exchange of messages between trading partners across firewalls have specialized security requirements, including trading partner authentication and authorization, as well as nonrepudiation. Specific security administration tasks depend on the type of protocol used. For more information about B2B integration security, see *Implementing Security with B2B Integration*.

In addition:

- For information about RosettaNet security, see "Configuring RosettaNet Security" in "Introduction" in *Implementing RosettaNet for B2B Integration.*
- For information about cXML security, see "Security" in "cXML Administration" in *Implementing cXML for B2B Integration*.

Application Integration Security

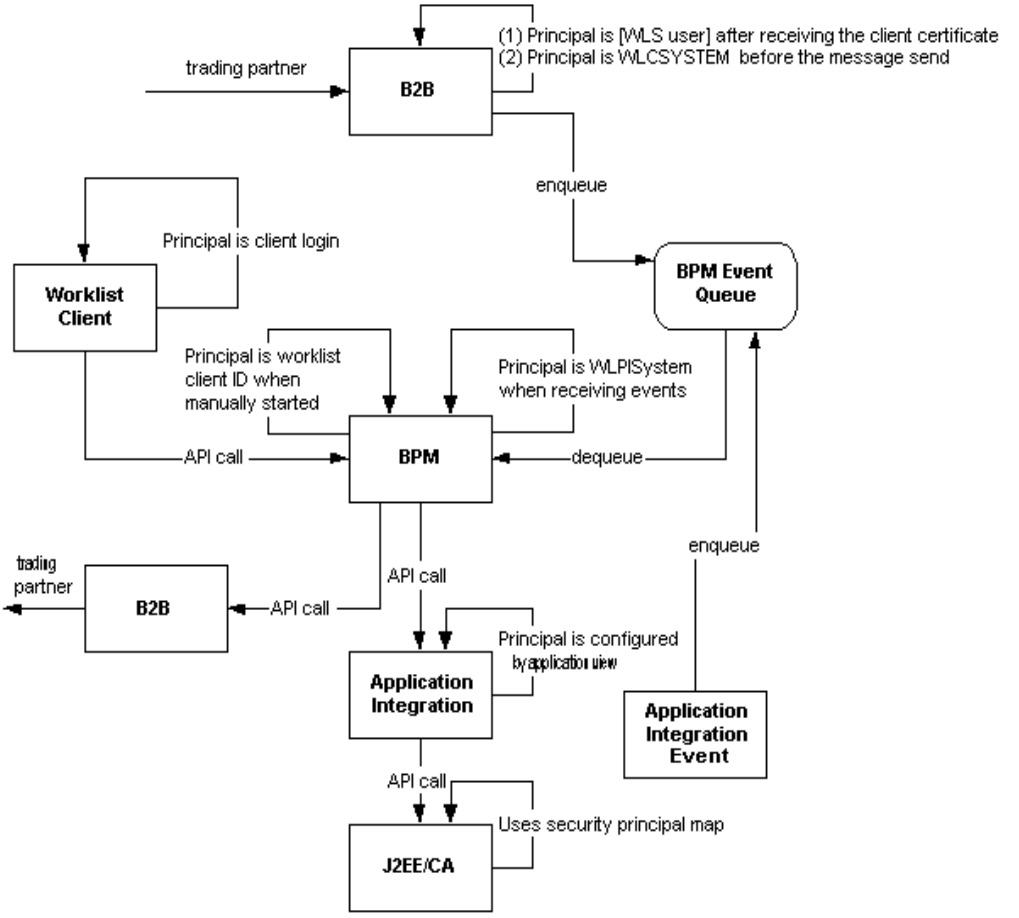
WebLogic Integration provides the following security mechanisms for those parts of an integration solution that are created and maintained with application integration functionality:

- 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 *Implementing 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 user or group. For more information, see "Deploying an Application View" in "Steps for Defining an Application View" in "Defining an Application View" in *Implementing Application Integration*.

WebLogic Server Security Principals Used in WebLogic Integration

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

Figure 3-1 WebLogic Server Security Principals Used in WebLogic Integration



- When started via an event (XML, application integration or B2B integration), a workflow runs as the WLPISystem.
- 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. It switches to the WLCSYSTEM 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 "Security Prinicipal Map" in *Programming the WebLogic J2EE Connector Architecture* in the WebLogic Server documenation set, at the following URL:

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

CHAPTER

4 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 the Pool Size of BPM Event Listener Message-Driven Beans
- Configuring the Number of Application Integration Asynchronous Request Threads
- Configuring Other 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/docs61/perform/index.html`

Configuring the Pool Size of BPM Event Listener Message-Driven Beans

The `wlpi-mdb.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. The default setting is 11 (5 unordered listeners plus 5 ordered listeners plus 1 time listener).

Use the MDBGenerator 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 whether you need more. See "Do You Have Enough Message-Driven Beans?" for more information.

Configuring the Number of Application Integration Asynchronous Request Threads

The `wlpi-ejb.jar` file contains the pool of session beans used to execute workflow instances, which are described in "Instance Beans" on page 1-9. You can configure the number of asynchronous request threads by adding the following line to the `wlai.properties` file:

`wlai.numAsyncServiceRequestProcessors=`*`numThreads`*

Here, *`numThreads`* is the number of asynchronous threads required. The default is 2.

Configuring Other 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 in the EJB Container" in *Programming WebLogic EJB* at the following URL:

`http://edocs.bea.com/wls/docs61/ejb/deploy.html`

Note: The pool size of BPM event listener message-driven beans should already be configured, as described in "Configuring the Pool Size of BPM Event Listener Message-Driven Beans" on page 4-3.

For each node in a WebLogic Integration cluster, complete the following steps:

1. Configure the cache size for the BPM workflow processor beans, which are described in "Workflow Processor Beans" on page 1-8.

   This setting should equal or exceed the size of the BPM event listener message-driven bean pool and it should also accommodate the anticipated workload from subworkflows or Worklist clients. The default setting is 100. The full name is `WorkflowProcessor` in `weblogic-ejb-jar.xml`.

2. Configure the cache size of the BPM template entity beans, which are described in "Template Beans" on page 1-9.

   This setting should equal or exceed the number of unique templates concurrently processed in the WebLogic Integration system. The default setting is 100. The full name is `TemplateDefinitionRO` in `weblogic-ejb-jar.xml`.

3. Configure the cache size of the BPM instance entity beans, which are described in "Instance Beans" on page 1-9.

   This setting should equal or exceed the number of workflow instance processors. The default setting is 100. The full name is `WorkflowInstance` in `weblogic-ejb-jar.xml`.

4. Configure the pool size of application view beans for application integration, which are described in "Application Integration Resources" on page 1-11.

   The default setting is 200, which is generally sufficient for most deployments. The full name is `com.bea.wlai.client.ApplicationView` in `weblogic-ejb-jar.xml`.

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 1-5.

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 4-1 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 sum total of 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/docs61/perform/WLSTuning.html`

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

  `http://edocs.bea.com/wls/docs61/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 1-6. For each node in a WebLogic Integration cluster, calculate the number of required execution threads based on the guidelines described in the following table.

Table 4-2 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 1 thread for every 4 messages per second, and then add 10 threads to the total.
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 of 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 "Thread Pool Size" in "Migrating WebLogic Server 6.0 Applications to WebLogic Server 6.1" in the *WebLogic Server 6.1 Release Notes* at the following URL:

`http://edocs.bea.com/wls/docs61/notes/migrate60to61.html`

You can find information on monitoring threads in "Do You Have Enough Threads?"

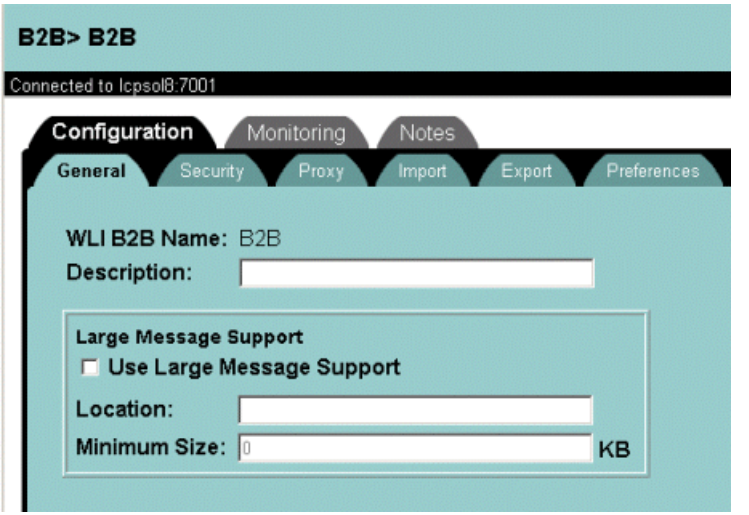
Configuring Resource Connection Pools for J2EE Connector Architecture Adapters

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 1-6. 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 by B2B conversations are too large to fit in memory, enable large message support on the B2B Console and restart the server. Figure 4-1 shows a portion of the console panel used for enabling large message support.

Figure 4-1 Large Message Support Area on B2B Console



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 VM 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/docs61/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 version of the JDK that is supplied with Weblogic Server supports two or three different JVM implementations. On Solaris systems, the Hotspot and the server JVM are supported. On Windows NT, the classic JVM is also supported.

The classic JVM is not recommended because it does not provide a JIT compiler. The server runs much more slowly (at least five times) with the classic JVM than with the Hotspot or server JVM.

The Hotspot and server JVM are identical except for the run-time compilation algorithms they use. (The Hotspot JVM is also known as the client JVM.)

The server JVM is more appropriate for use with Weblogic Integration. 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.

There is a known bug that can occur with the server JVM, which causes the JVM to allocate all the available memory in the system, regardless of the heap size specified (see Sun Bug ID 4484370). If this happens, you should use Hotspot JVM, instead.

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 cut into two parts: the new or Eden heap, and the tenured heap.

All new objects are created in the Eden heap. They are moved to the tenured heap only after surviving garbage collection from the Eden heap. The tenured heap is not collected as often as the Eden heap, and the collection operation for it is a lot more expensive than collection for the Eden heap. A rule of thumb is that the Eden 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 Eden 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 Eden heap. (For this reason, this approach is sometimes referred to as delayed garbage collection).

With a global heap of 512Mb, a reasonable size for the Eden heap is 128Mb, as shown in the following option setting:

```
-XX:NewSize=128m –XX:MaxNewSize=128m
```

Garbage collection in the Eden heap is generational. Objects are created initially in a part of the Eden heap that contains only the young generation. Every time an object is considered for collection, but is still being used, its generation number is incremented and the object is copied to a survivor space in the Eden heap. After a number of generations, the object is declared old and moved to the tenured space. The Eden heap contains two survivor spaces, only one of which is used at a time. The number of generations that must be reached before objects are moved in the tenured heap is determined dynamically by the JVM to keep the survivor spaces half-full.

The size of the survivor spaces can be specified as a ratio of the Eden heap. If survivor spaces are too small, copying collection overflows directly into the old generation. If survivor spaces are too large, they are uselessly empty.

The recommended value for the survivor ratio is 2. When this value is used, each survivor space is half the size of the young generation. Because there are two survivor spaces, the space for the young generation is ½ the size of the Eden heap, and each survivor space is ¼ the size of the Eden heap. Use the following option setting to specify a survivor ratio of 2:

```
-XX:SurvivorRatio=2
```

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 AI 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 *BEA WebLogic Server Administration Guide* at the following URL:

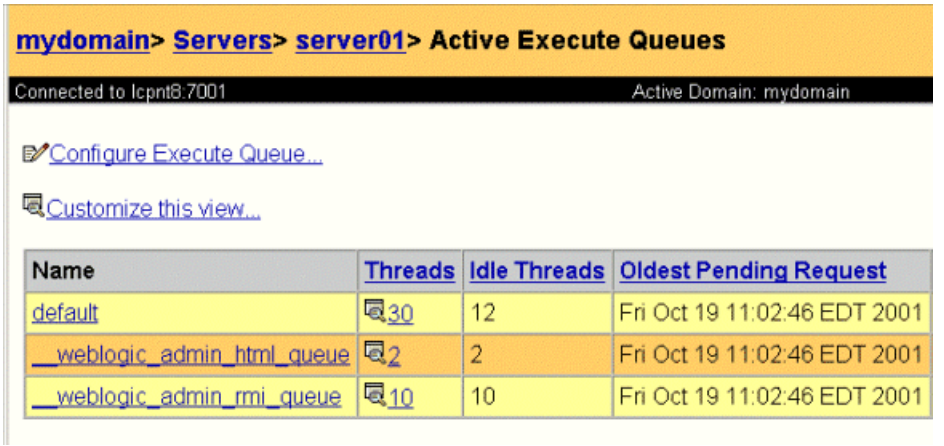
`http://edocs.bea.com/wls/docs61/adminguide/monitoring.html`

Do You Have Enough Threads?

In the left frame of the Weblogic Server Administration Console, select `Servers > server_name`. In the right frame, select the Monitoring tab.

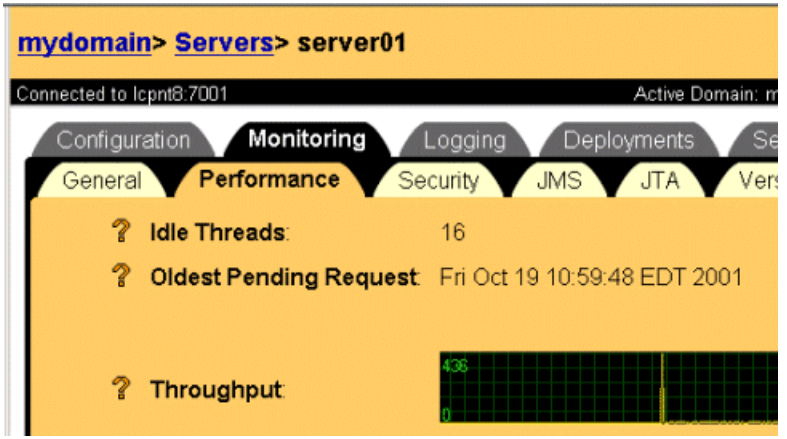
The General tab allows you to display a table with information on the execute queues, including the number of idle threads. Figure 4-2 shows how the WebLogic Server Administration Console displays information about active queues.

Figure 4-2 Active Execute Queues Table



Also under Monitoring, there is a Performance tab. This tab displays three graphs: Throughput, Queue Length, and Memory Usage. Above the graphs is the Idle Threads field. If the number shown in the Idle Threads field is sometimes zero, you need more threads. The parameter that controls the number of threads is `ThreadPoolSize`. The `ThreadPoolSize` parameter is set separately for each server. Figure 4-3 shows how the WebLogic Server Administration Console displays performance information.

Figure 4-3 Server Performance Information



To add more threads, select the default queue and specify the thread count. Figure 4-4 shows how the WebLogic Server Administration Console displays execute queue information.

Figure 4-4 Execute Queue Table

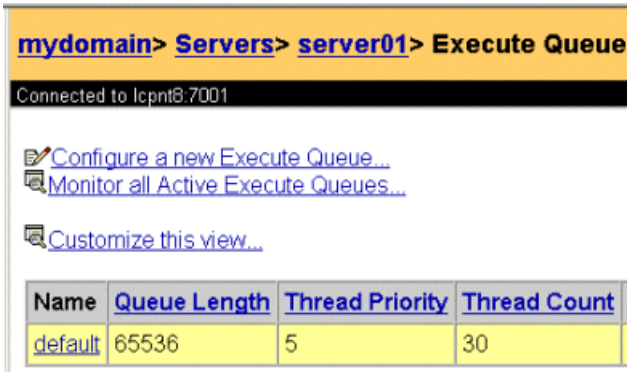
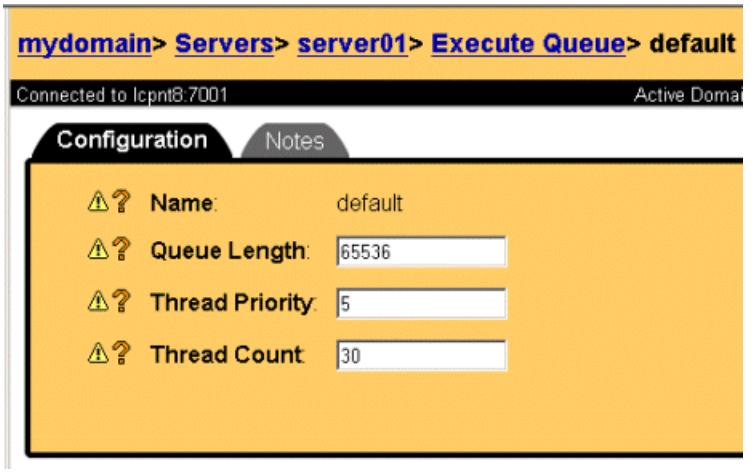


Figure 4-5 shows the WebLogic Server Administration Console tab used to specify the thread count.

Figure 4-5 Default Execute Queue Configuration



On Solaris, you can also determine whether changing the number of threads improves performance by running the `mpstat` command at comparable load levels before and after changing the setting. A drop in the number of context switches suggests that performance has improved.

How Many Transactions Are Occurring?

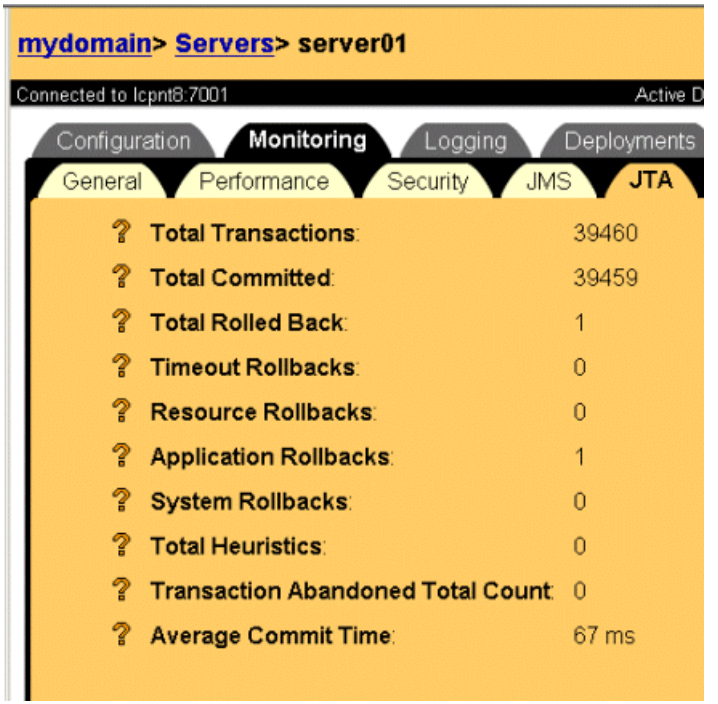
To display the number of transactions of various types, select your server name in the Weblogic Server Administration Console. In the right frame, select the Monitoring tab, then the JTA tab. Select `Monitor all instances`.

Some transactions are associated with the BPM framework, and you cannot change them. You may choose to change transaction types or combine transactions for those transactions associated with your applications.

```
Server > Monitoring > JTA
```

Figure 4-6 shows the WebLogic Server Administration Console tab used to monitor transactions.

Figure 4-6 JTA Monitoring Tab



Do You Have Enough JDBC Connections?

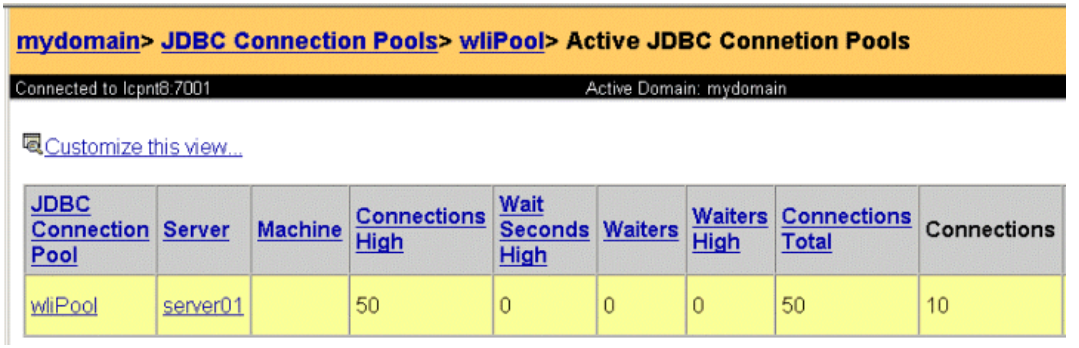
JDBC connections are connections to your database, made available so that individual threads do not suffer performance problems caused by getting a new connection every time access to the database is required. You may have multiple pools of JDBC connections. It is important that each pool has enough connections so that no thread has to wait long for a connection.

In the left frame, select `Services` > `JDBC` > `Connection Pools`. Select a pool and then select `Monitor active connection pools`.

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

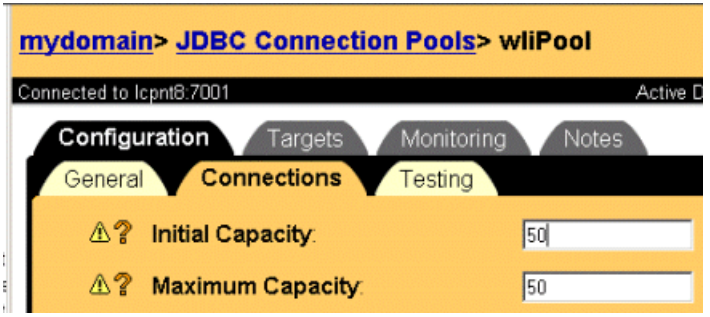
may prove useful under similar situations or when load increases slightly. Figure 4-7 shows the WebLogic Server Administration Console window used to monitor active connection pools.

Figure 4-7 Active JDBC Connection Pools



To modify connection pool configuration, go to `Services` > `JDBC` > `Connection Pools` > `wliPool`, then select Connections and set the values for the Initial Capacity and Maximum Capacity fields to the same number. Figure 4-8 shows the WebLogic Server Administration Console tab used to set initial and maximum capacity.

Figure 4-8 Connection Pool Configuration



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.

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 4-1 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 4-2 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?

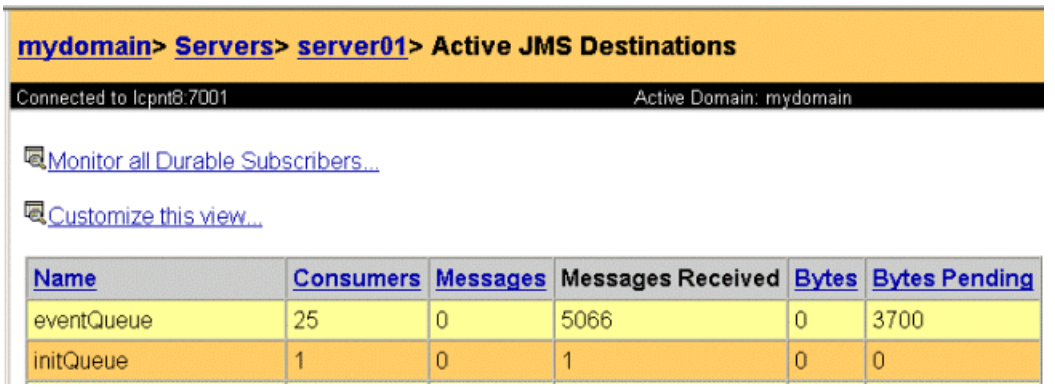
To display information on message-driven beans, select your server by name in the Weblogic Server Administration Console. Then, in the right frame, select the Monitoring tab, followed by the JMS tab. Select `Monitor all Active JMS Servers > Active JMS Destinations > JMSServer-0`.

Look at the queue length for `eventQueue`. If the number is often more than just a few, more queuing is occurring than is desirable for good performance. In this case, adding more MDBs helps performance. Figure 4-9 shows the WebLogic Server Administration Console tab used to monitor the event queue.

`Server > Monitoring > JMS > Monitor all Active JMS Servers > Active JMS Destinations > JMSServer-0`

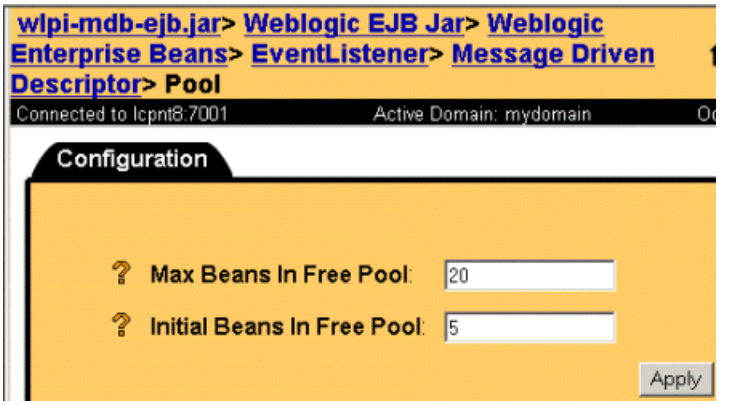
`EventQueue – Messages / Messages Received`

Figure 4-9 Event Queue Monitoring



To change the number of MDBs, select `EJB > wlpi-mdb-ejb.jar` in the left frame of the Weblogic Server Administration Console. Then, in the right frame, select `Edit EJB Descriptor`. A new window is displayed showing the `Max Beans in Free Pool` and `Initial Beans in Free Pool` parameters. Edit the value of the `Max Beans in Free Pool` parameter. You must reboot Weblogic Server for this change to take effect. Figure 4-10 shows the WebLogic Server Administration Console tab used to edit the `Max Beans in Free Pool` parameter.

Figure 4-10 Configuring MDBs



How Many of Each Type of Bean Does My System Have?

Use the WebLogic Server Administration Console to display information bean types and quantities. In the left frame, select a particular EJB jar. In the right frame, select the Monitoring tab and the type of bean to be displayed. For example, to display information about stateful session beans, select `Monitor all Stateful Session Beans`.

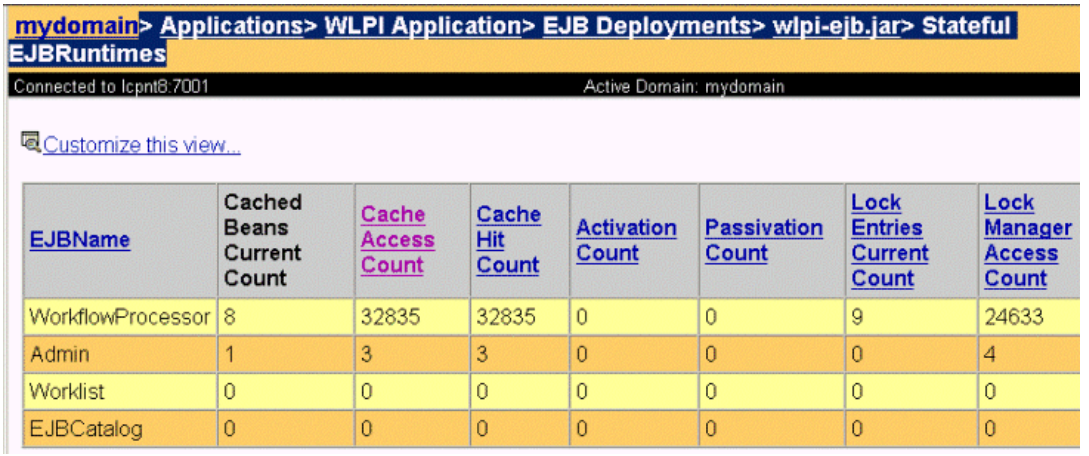
To modify the display of information, select `Customize this view.` You can add or delete columns and change the sort order. Add all the remaining columns. (Highlight the columns, click the arrow to move them to the right, and then press Apply.) The following columns are of particular interest:

```
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`, and the jar files for your application-specific EJBs. Figure 4-11, Figure 4-12, and Figure 4-13 show portions of the windows in which information for stateful, entity, and message-driven beans is displayed.

```
Applications > WLPI Application > EJB Deployment > wlpi-ejb.jar >
Stateful EJBRuntimes
```

Figure 4-11 Stateful Bean Information



mydomain> Applications> WLPI Application> EJB Deployments> wlpi-ejb.jar> Stateful EJBRuntimes

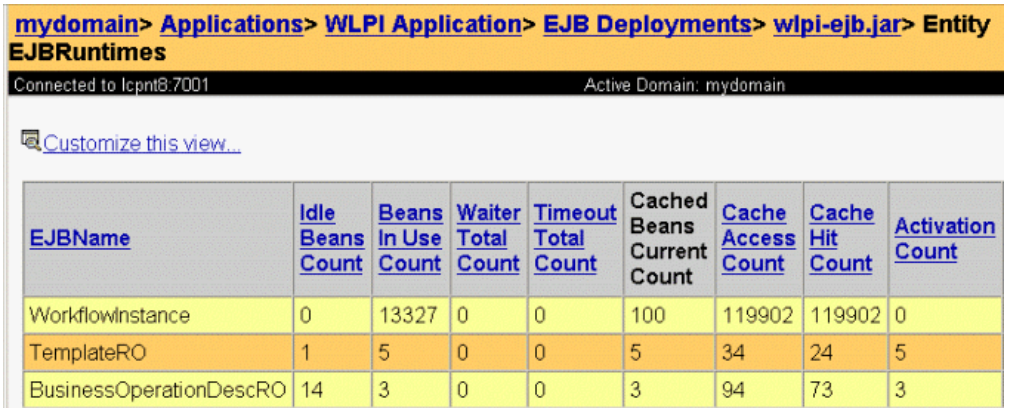
Connected to lcpnt8:7001 Active Domain: mydomain

Customize this view...

EJBName	Cached Beans Current Count	Cache Access Count	Cache Hit Count	Activation Count	Passivation Count	Lock Entries Current Count	Lock Manager Access Count
WorkflowProcessor	8	32835	32835	0	0	9	24633
Admin	1	3	3	0	0	0	4
Worklist	0	0	0	0	0	0	0
EJBCatalog	0	0	0	0	0	0	0

```
Applications > WLPI Application > EJB Deployment > wlpi-ejb.jar >
Entity EJBRuntimes
```

Figure 4-12 Entity Bean Information



mydomain> Applications> WLPI Application> EJB Deployments> wlpi-ejb.jar> Entity EJBRuntimes

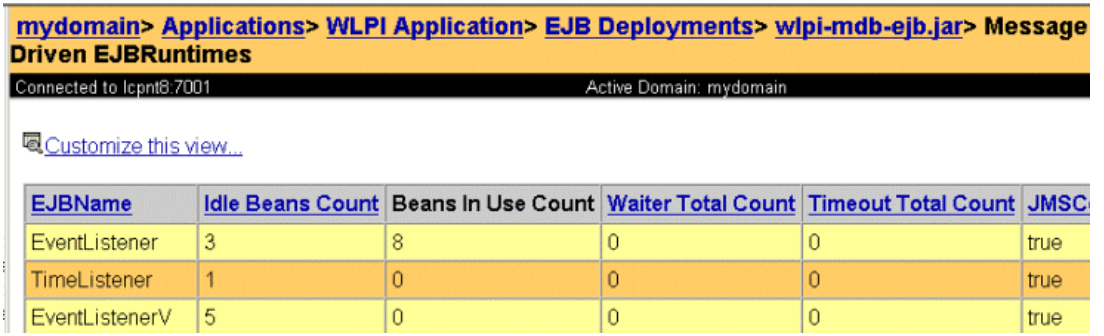
Connected to lcpnt8:7001 Active Domain: mydomain

Customize this view...

EJBName	Idle Beans Count	Beans In Use Count	Waiter Total Count	Timeout Total Count	Cached Beans Current Count	Cache Access Count	Cache Hit Count	Activation Count
WorkflowInstance	0	13327	0	0	100	119902	119902	0
TemplateRO	1	5	0	0	5	34	24	5
BusinessOperationDescRO	14	3	0	0	3	94	73	3

`Applications > WLPI Application > EJB Deployment > wlpi-mdb-ejb.jar > Message Driven EJBRuntimes`

Figure 4-13 MDB Information



mydomain> Applications> WLPI Application> EJB Deployments> wlpi-mdb-ejb.jar> Message Driven EJBRuntimes

Connected to lcpnt8:7001 Active Domain: mydomain

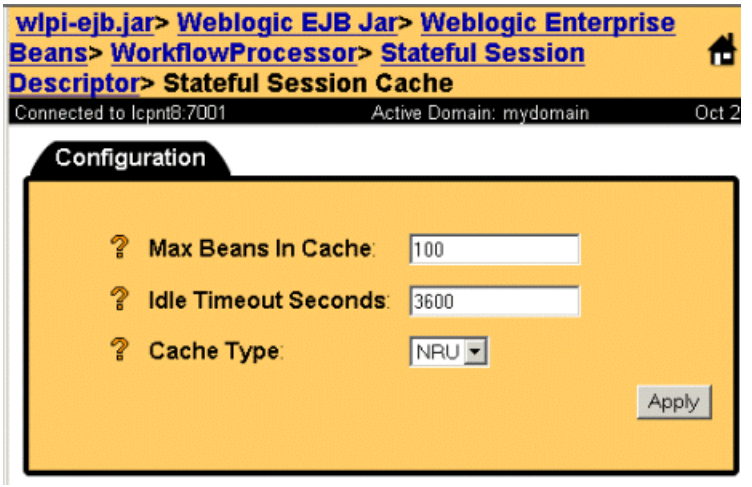
Customize this view...

EJBName	Idle Beans Count	Beans In Use Count	Waiter Total Count	Timeout Total Count	JMSC
EventListener	3	8	0	0	true
TimeListener	1	0	0	0	true
EventListenerV	5	0	0	0	true

If a system message concerning *cache full* is displayed, increase the corresponding bean's `Max Beans in Cache` parameter 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. Display the bean in the WebLogic Server Administration Console and click the `Edit EJB Descriptor` link. Figure 4-14 shows the WebLogic Server Administration Console tab used to edit the `Idle Timeout Seconds` parameter.

Figure 4-14 Idle Timeout Configuration



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 a 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. You can monitor logs and message statistics using the WebLogic Integration B2B Console tabs shown in Figure 4-15 and Figure 4-16.

Figure 4-15 Monitoring B2B Logs

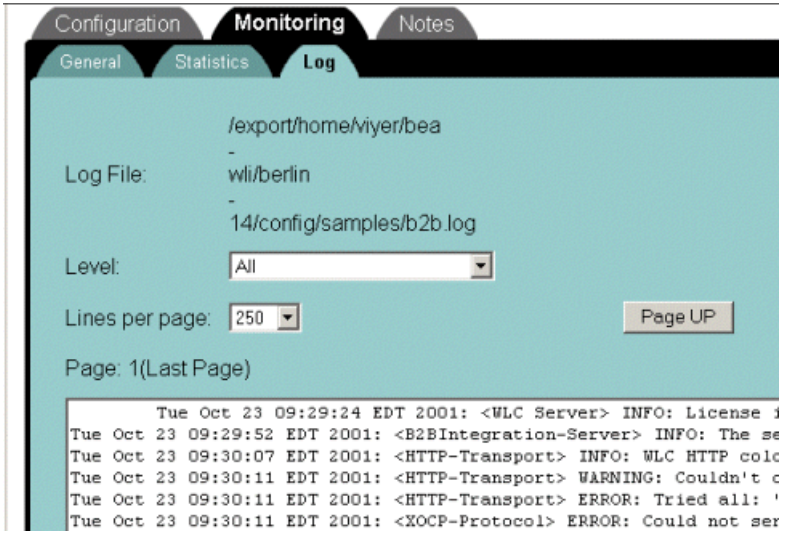
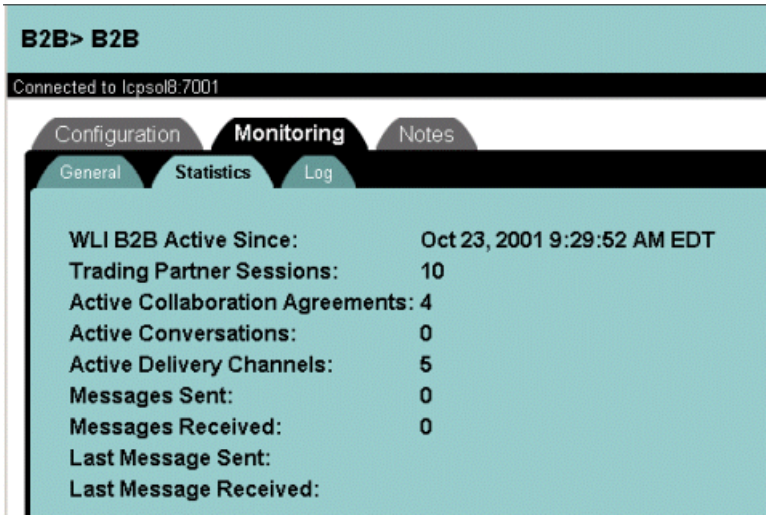


Figure 4-16 Monitoring B2B Statistics



Monitoring and Tuning AI Performance

This section provides information about:

- Monitoring and Tuning Application View Connections
- Monitoring and Tuning Queues for Asynchronous Services
- Enabling Transactions and Persistence in Asynchronous Service Request/Response Handling for JMS

Monitoring and Tuning Application View Connections

To check whether you have sufficient connections available for your application view, start the Weblogic Server Administration Console and select `Deployments > Connectors`.

Select the connection factory deployed for your application view, which is named using the following format:

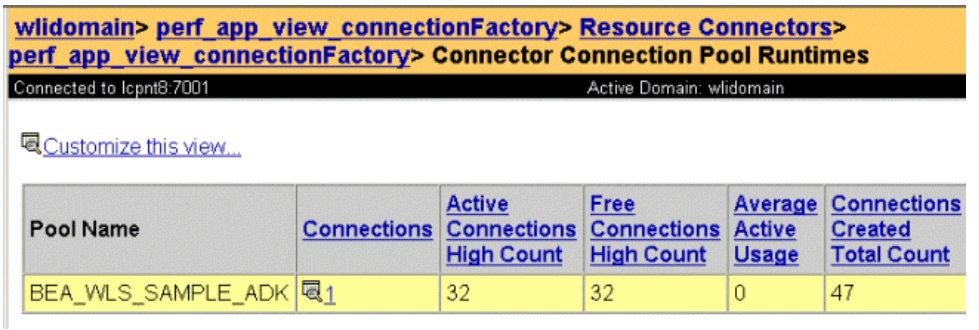
*ApplicationViewName*_connectionFactory.

Select the Monitoring tab and click Monitor all Connector Connection Pool Runtimes...

The connections to the EIS defined in your application view are displayed. 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.

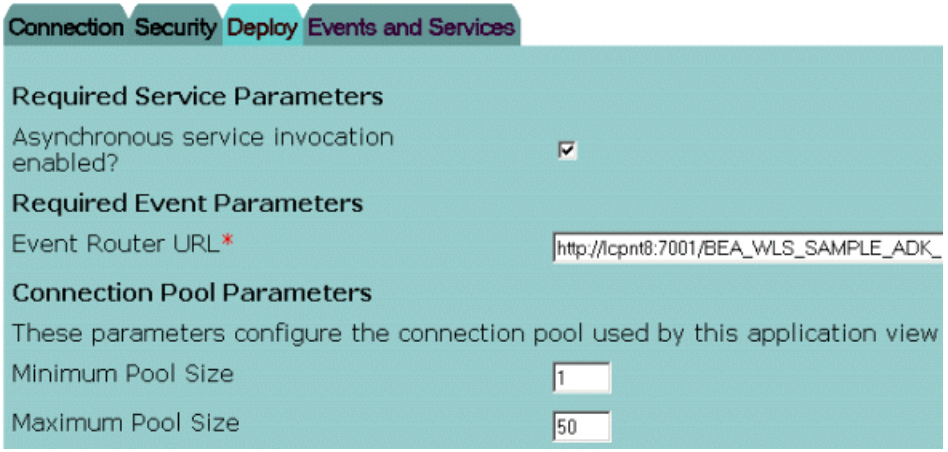
Look at the number of connections; is it close to the total number of connections configured for this pool? Is the Active Connections High Count value equal to the total number of connections configured for this pool? Either of these is a sign that more connections might prove useful under similar situations or when load increases slightly. Figure 4-17 shows the WebLogic Server Administration Console tab used to monitor connections.

Figure 4-17 Monitoring Application View Connection



To view or modify your maximum connections for your application view, go to the Application View Console, select your application view, and select the Deploy tab. The Maximum Pool Size value shows the maximum number of connections. Figure 4-18 shows the Application View Console tab used to monitor the maximum pool size.

Figure 4-18 Monitoring Maximum Pool Size

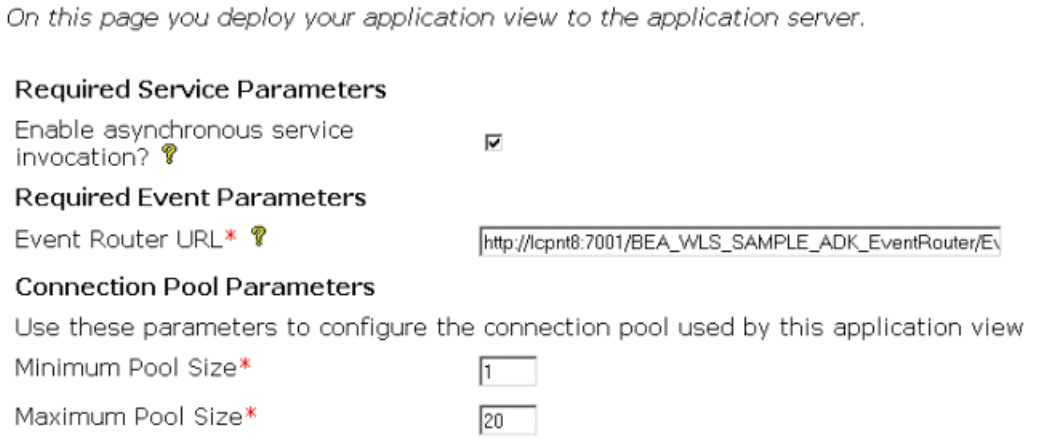


To modify this value, perform the following steps:

1. Click Undeploy if the Application View is currently deployed.
2. When the Application View is undeployed, click Edit.
3. Click Continue.
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.

Figure 4-19 shows the Application View Console tab used to edit the maximum pool size.

Figure 4-19 Modifying Maximum Pool size



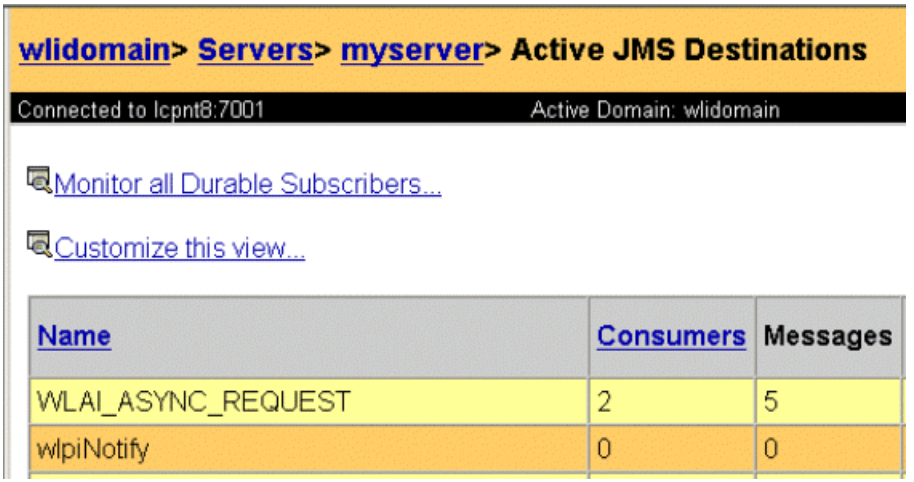
Monitoring and Tuning Queues for Asynchronous Services

When asynchronous services are invoked, the service responses are queued in the `WLAI_ASYNC_REQUEST` JMS queue. The consumers for this queue are worker threads, which are set to 2 by default. As the number of concurrent invocations increases, asynchronous service responses can start to fill up the `WLAI_ASYNC_REQUEST` queue. To determine whether the queue is filling up, display the Active JMS Destinations window:

```
Server > Monitoring > JMS > Monitor all Active JMS Servers > Active
JMS Destinations
```

Check the number of messages for `WLAI_ASYNC_REQUEST`. Figure 4-20 shows the WebLogic Server Administration Console tab used to monitor the number of messages.

Figure 4-20 Monitoring Messages for WLAI_ASYNC_REQUEST



If messages are queuing up, increase the number of worker threads. The more you have, the more asynchronous service invocations the server can support. This parameter is set by editing the following line in the `wlai.properties` file:

```
wlai.numAsyncServiceRequestProcessors=2
```

The number of processors is 2 by default, so you will likely need to increase this number. Use the following formula for calculating the number of processors that gives the theoretical maximum throughput of asynchronous requests and responses:

```
num_async_processors = avg_clients * avg_services/sec *
avg_service_duration
```

The following table describes the values you must provide for each formula element.

num_async_processors	The value to specify for `wlai.numAsyncServiceRequestProcessors`.
avg_clients	The average number of clients you expect to be invoking services asynchronously.
avg_services/sec	The average number of asynchronous service invocations you expect EACH client to initiate per second.
avg_service_duration	The average time a given service invocation will take, in seconds.

These averages are sometimes difficult to calculate, so you may have to estimate a value and then observe the results. In general, if your `WLAI_ASYNC_REQUEST` queue is filling up, you should add more processors.

Enabling Transactions and Persistence in Asynchronous Service Request/Response Handling for JMS

By default, transactions and persistence are disabled. You can enable them as required. Turn on transactions by adding or modifying the following line in the `wlai.properties` file:

```
wlai.jms.asyncServiceTransFlag=true
```

To enable or disable persistence, modify the `WLAI_JMSConnectionFactory` JMS Connection factory in the WebLogic Server Administration Console by changing the default delivery mode to `Persistent` or `NonPersistent`.

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/docs61/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 4-3 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
- Tuning Cloudscape 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, the only version of the Oracle database supported by Weblogic Integration 2.1

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`

where *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 "foo"
```

is roughly equivalent to the following query:

```
SELECT name, value FROM v$parameter WHERE name LIKE '%foo%';
```

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 tuning options in this section will have a positive effect on performance and parameter values may need to be empirically derived.

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). Thus, if you are not using these Oracle features, then the processes are wasted resources. Turn off these processes until they are actually needed. This can be done by modifying the `init.ora` file. The safest approach is to comment 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 4-4 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 4-5 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 4-5 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.

Tuning Cloudscape Databases

BEA provides Cloudscape support only for development on Windows platforms. We recommend using a database other than Cloudscape for production for the following reasons:

- With high loads, Cloudscape performance is very poor.
- Cloudscape is more prone to deadlocks in a multiCPU, multiple-client deployment.
- Migration of earlier WebLogic Integration versions to WebLogic Integration Release 2.1 is not supported when the underlying database is Cloudscape.
- Bundling of Cloudscape might be discontinued in future WebLogic Integration releases.

Index

R

S

T

W